Samson Abramsky
Cyril Gavoille
Claude Kirchner
Friedhelm Meyer auf der Heide
Paul G. Spirakis (Eds.)

LNCS 6198
ARCoSS

Automata, Languages and Programming

37th International Colloquium, ICALP 2010
Bordeaux, France, July 2010
Proceedings, Part I

Part I

Springer

Lecture Notes in Computer Science 6198

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Advanced Research in Computing and Software Science
Subline of Lectures Notes in Computer Science

Samson Abramsky
Cyril Gavoille
Claude Kirchner
Friedhelm Meyer auf der Heide
Paul G. Spirakis (Eds.)

Automata, Languages and Programming

37th International Colloquium, ICALP 2010
Bordeaux, France, July 6-10, 2010
Proceedings, Part I

Springer

Volume Editors

Samson Abramsky
Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD, UK
E-mail: samson.abramsky@comlab.ox.ac.uk

Cyril Gavoille
Université de Bordeaux (LaBRI) & INRIA
351, cours de la Libération, 33405 Talence Cedex, France
E-mail: gavoille@labri.fr

Claude Kirchner
INRIA, Centre de Recherche Bordeaux – Sud-Ouest
351 cours de la Libération, 33405 Talence Cedex, France
E-mail: claude.kirchner@inria.fr

Friedhelm Meyer auf der Heide
Heinz Nixdorf Institute, University of Paderborn
Fürstenallee 11, 33102 Paderborn, Germany
E-mail: fmadh@upb.de

Paul G. Spirakis
University of Patras and RACTI
26500 Patras, Greece
E-mail: spirakis@cti.gr

Library of Congress Control Number: 2010929577

CR Subject Classification (1998): I.2, F.2, F.1, C.2, H.3, G.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-642-14164-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-14164-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

ICALP 2010, the 37th edition of the International Colloquium on Automata, Languages and Programming was held July 6-10, 2010 in Bordeaux, France. ICALP is a series of annual conference of the European Association for Theoretical Computer Science (EATCS) which first took place in 1972, organized by Maurice Nivat and his colleagues in Paris. This year, the program consisted of the established track A, focusing on Algorithms, Complexity and Games, chaired by Paul G. Spirakis; Track B, focusing on Logic, Semantics, Automata and Theory of Programming, chaired by Samson Abramsky; Track C focusing this year on Foundations of Networked Computation: Models, Algorithms and Information Management, chaired by Friedhelm Meyer auf der Heide.

The three Program Committees received a total of 389 submissions: 222 for Track A, 114 for Track B and 53 for Track C, written by authors from 45 different countries. Of these, 60, 30 and 16, respectively, were selected for inclusion in the scientific program. Each paper got on average 3.5 referee reports.

The Program also included six invited talks by Pierre Fraigniaud (CNRS and Univ. Paris Diderot), Jean Goubault-Larrecq (ENS Cachan and LSV), Burkhard Monien (Univ. Paderborn), Joel Ouaknine (Oxford Univ. Computing Lab.), Roger Wattenhofer (ETH Zurich), and Emo Welzl (ETH Zurich).

These 112 contributed and invited papers are presented in two proceedings volumes. The first contains the contributed papers of Track A and the invited talks of Burkhard Monien and Emo Welzl. The second volume contains the contributed papers of Tracks B and C as well as the invited talks of Pierre Fraigniaud, Jean Goubault-Larrecq, Joel Ouaknine and Roger Wattenhofer.

The day before the main conference, five satellite workshops were held:
- AlgoGT : Workshop on Algorithmic Game Theory: Dynamics and Convergence in Distributed Systems
- DYNAS 2010: International Workshop on DYnamic Networks: Algorithms and Security
- ALGOSENSORS 2010: International Workshop on Algorithmic Aspects of Wireless Sensor Networks
- SDKB 2010: Semantics in Data and Knowledge Bases
- TERA-NET: Towards Evolutive Routing Algorithms for Scale-Free/Internet-Like Networks.

We wish to thank all the authors of submitted papers, all the members of the three Program Committees for their scholarly effort and all 737 referees who assisted the Program Committees in the evaluation process.

We are very pleased to thank INRIA for organizing the conference, LaBRI for their collaboration, and the sponsors (Conseil Rgional d'Aquitaine, Communauté Urbaine de Bordeaux, CEA, CNRS via the GDR IM, Total) for their strong support. We are also very grateful to Ralf Klasing for chairing the workshop

organization and to all the members of the Organizing Committee: Laëtitia Grimaldi, Alice Rivière, Nicolas Bonichon, Pierre Casteran, Lionel Eyraud-Dubois and Frédéric Mazoit.

It is also our great pleasure to acknowledge the use of the EasyChair conference management system, which was of tremendous help in handling the submission and refereeing processes as well as in intelligently assisting us in the design of the final proceedings.

May 2010

Samson Abramsky
Cyril Gavoille
Claude Kirchner
Friedhelm Meyer auf der Heide
Paul G. Spirakis

Organization

Conference Chairs

Cyril Gavoille	University of Bordeaux (LaBRI) and INRIA, Bordeaux, France
Claude Kirchner	INRIA, Bordeaux, France

Programme Chairs

Samson Abramsky	Oxford University, UK
Friedhelm Meyer auf der Heide	University of Paderborn, Germany
Paul G. Spirakis	University of Patras and RACTI, Greece

Workshops Chair

Ralf Klasing	CNRS, University of Bordeaux (LaBRI) and INRIA, France

Program Committee

Track A

Dimitris Achlioptas	University of Santa Cruz, USA
Kwstis Daskalakis	MIT, USA
Anuj Dawar	Cambridge University, China SAR
Xiaotie Deng	City University of Hong Kong, France
Philippe Flajolet	INRIA
Michele Flammini	University of L'Aquila, Italy
Oscar Ibarra	UCSB (Santa Barbara), USA
Giuseppe Italiano	University of Rome 2, Italy
Kazuo Iwama	University of Kyoto, Japan
Klaus Jansen	University of Kiel, Germany
Elias Koutsoupias	University of Athens, Greece
Ludek Kucera	Charles University, Czech Republic
Daniel Marx	Budapest University, Hungary
Burkhard Monien	University of Paderborn, Germany
Amin Coja Oghlan	University of Edinburgh, UK
Tonyann Pitassi	University of Toronto, Canada
John Reif	Duke University, USA
Jose Rolim	University of Geneva, Switzerland

Paul G. Spirakis	University of Patras and RACTI (Chair), Greece
Leslie Valiant	Harvard University, USA
Emo Welzl	ETH, Switzerland
Gerhard Woeginger	University of Eindhoven, The Netherlands

Track B

Samson Abramsky	Oxford University, UK, (Chair)
Luca Aceto	University of Rejkjavik, Iceland
Lars Birkedal	IT University of Copenhagen, Denmark
Mikolaj Bojanczyk	University of Warsaw, Poland
Patricia Bouyer	CNRS, LSV Cachan, France
Josée Desharnais	University of Laval, Canada
Gilles Dowek	Ecole Polytechnique and INRIA, France
Manfred Droste	University of Leipzig, Germany
Peter Dybjer	University Chalmers, Sweden
Jose Felix Costa	University of Lisbon, Portugal
Phokion Kolaitis	IBM Almaden, USA
Ugo Dal Lago	University of Bologna, Italy
Daniel Leivant	University of Indiana, USA
Andrzej Murawski	Oxford University, UK
Filip Murlak	University of Warsaw, Poland
Flemming Nielsen	University of Copenhagen, Denmark
Dominique Perrin	University of Paris Est, France
Alex Rabinovich	University of Tel Aviv, Israel
Lutz Schröder	DFKI Bremen, Germany
Ian Stark	University of Edinburgh, UK

Track C

Debora Donato	Yahoo! Research Barcelona, Spain
Faith Ellen	University of Toronto, Canada
Phil Gibbons	Intel Research Pittsburgh, USA
Rob van Glabbeek	Stanford University and National ICT Australia
Monika Henzinger	EPFL Lausanne, Switzerland
Christos Kaklamanis	University of Patras, Greece
Fabian Kuhn	MIT, USA
Miroslaw Kutylowski	Wroclaw University of Technology, Poland
Christian Lengauer	University of Passau, Germany
Stefano Leonardi	Sapienza University of Rome, Italy
Friedhelm Meyer auf der Heide	University of Paderborn, Germany (Chair)
Dusko Pavlovic	Oxford University and Kestrel Institute, UK
Andrzej Pelc	Université du Qubec en Outaouais, Canada
Giuseppe Persiano	University of Salerno, Italy

Frank Pfenning	CMU, USA
Geppino Pucci	University of Padova, Italy
Christian Scheideler	University of Paderborn, Germany
Nir Shavit	Tel Aviv University, Israel
Berthold Vöcking	RWTH Aachen, Germany
Gerhard Weikum	MPI-Saarbrücken, Germany

Organizing Committee

Laëtitia Grimaldi	INRIA, Bordeaux, France (Conference Secretariat)
Nicolas Bonichon	University of Bordeaux (LaBRI) and INRIA, France
Pierre Casteran	University of Bordeaux (LaBRI), France
Lionel Eyraud-Dubois	INRIA and University Bordeaux (LaBRI), France
Frédéric Mazoit	University of Bordeaux (LaBRI), France
Alice Rivière	INRIA, Bordeaux, France

Sponsoring Institutions

CEA (Commissariat à l'énergie atomique et aux énergies alternatives)
Communauté Urbaine de Bordeaux
Conseil Régional d'Aquitaine
GDR Informatique Mathmatique (CNRS)
INRIA
Total

Referees for Track A

Mohammad Abam	Tugkan Batu	Vincenzo Bonifaci
David Adjiashvili	Paul Beame	Endre Boros
Pankaj Aggarwal	Luca Becchetti	Prosenjit Bose
Ali Akhavi	Siavosh Benabbas	Xavier Boyen
Susanne Albers	Eli Ben-Sasson	Andreas Brandstadt
Kazuyuki Amano	Dietmar Berwanger	Mark Braverman
Andris Ambainis	Davide Bilo'	Patrick Briest
Boris Aronov	Vittorio Bilo'	Yves Brise
Abdullah Arslan	Markus Blaeser	Joshua Brody
Vincenzo Auletta	Guy Blelloch	Tian-Ming Bu
Per Austrin	Johannes Bloemer	Niv Buchbinder
Evripidis Bampis	Johannes Blömer	Harry Buhrman
Nikhil Bansal	Hans L. Bodlaender	Andrei Bulatov
Johann Barbier	Tom Bohman	Costas Busch
David Mix Barrington	Beate Bollig	Jaroslaw Byrka

Sergio Cabello
Christian Cachin
Jin-Yi Cai
Yang Cai
Ioannis Caragiannis
Benjamin Carle
Marco Cesati
Timothy M. Chan
Harish Chandran
Arkadev Chattopadhyay
Ioannis Chatzigiannakis
Chandra Chekuri
Hong-Bin Chen
Ning Chen
Xi Chen
Xujin Chen
Hong-Bin Chen
Andrew Childs
Tobias Christ
Giorgos Christodoulou
Vincent Conitzer
Stephen Cook
Colin Cooper
Jose Correa
Peter Damaschke
Samir Datta
Matei David
Ronald de Wolf
Erik D. Demaine
Camil Demetrescu
Luc Devroye
Gabriele Di Stefano
Ilias Diakonikolas
Martin Dietzfelbinger
Shlomi Dolev
Frederic Dorn
Laurent Doyen
Dominic Dumrauf
Martin Dyer
Alon Efrat
Charilaos Efthymiou
Omer Egecioglu
Friedrich Eisenbrand
Mourad El Ouali
Robert Elsaesser
Ioannis Z. Emiris
Matthias Englert
Leah Epstein
Jeff Erickson
Thomas Erlebach
Alex Fabrikant
Jittat Fakcharoenphol
Angelo Fanelli
Qizhi Fang
Michael Fellows
Stefan Felsner
Amos Fiat
Irene Finocchi
Matthias Fischer
Holger Flier
Luca Forlizzi
Dimitris Fotakis
Paolo Giulio Franciosa
Andrea Francke
Daniele Frigioni
Hiroshi Fujiwara
Martin Gairing
David Gamarnik
Bernd Gärtner
William Gasarch
Dmitry Gavinsk
Heidi Gebauer
Joachim Gehweiler
Craig Gentry
Loukas Georgiadis
Konstantinos Georgiou
Christos Gkantsidis
Andreas Goerdt
Leslie Ann Goldberg
Shafi Goldwasser
Michael Goodrich
Fabrizio Grandoni
Venkatesan Guruswami
Magnus Halldorsson
Sean Hallgren
Xin Han
Moritz Hardt
Simai He
Christoph Helmberg
John Hitchcock
Jan Hladky
Martin Hoefer
Michael Hoffmann
Thomas Holenstein
Steve Homer
Stefan Hougardy
Tomas Hruz
Lei Huang
Benoit Hudson
Chim Hung
Paul Hunter
Thore Husfeldt
Nicole Immorlica
Takehiro Ito
Chuzo Iwamoto
Gerold Jaeger
Martin Jaggi
Kamal Jain
Tao Jiang
David Juedes
Valentine Kabanets
Satyen Kale
Sampath Kannan
Mamadou Kanté
Marc Kaplan
Alexis Kaporis
Juha Kärkkäinen
Andreas Karrenbauer
Telikepalli Kavitha
Akinori Kawachi
Jun Kawahara
Akitoshi Kawamura
Ken-ichi Kawarabayashi
Julia Kempe
Barbara Kempkes
Iordanis Kerenidis
Sanjeev Khanna
Aggelos Kiayias
Felix Klaedtke
Robert Kleinberg
Lasse Kliemann
Sebastian Kniesburges
Johannes Koebler
Jochen Koenemann
Robert Koenig

Stavros Kolliopoulos
Spyros Kontogiannis
Guy Kortsarz
Takeshi Koshiba
Annamaria Kovacs
Daniel Kral
Evangelos Kranakis
Dieter Kratsch
Stefan Kratsch
Robert Krauthgamer
Stephan Kreutzer
Piotr Krysta
Raghav Kulkarni
Eduardo Laber
Oded Lachish
Michael Langberg
Luigi Laura
Ron Lavi
Francois Le Gall
Lap-Kei Lee
Troy Lee
Pierre Leone
Hing Leung
Asaf Levin
Guojun Li
Vasiliki Liagkou
Nutan Limaye
Yi-Kai Liu
Daniel Lokshtanov
Pinyan Lu
Avner Magen
Peter Mahlmann
Mohammad Mahmoody
David F. Manlove
Alberto Marchetti-Spaccamela
Monaldo Mastrolilli
Claire Mathieu
Marios Mavronicolas
Elvira Mayordomo
Andrew McGregor
Frank McSherry
Giovanna Melideo
Ricardo Menchaca
Henning Meyerhenke
Adam Meyerson
Dimitrios Michail
Othon Michail
Nikola Milosavljevic
Vahab Mirrokni
Shuichi Miyazaki
Gianpiero Monaco
Morteza Monenizadeh
Hiroki Morizumi
Luca Moscardelli
Philippe Moser
Robin Moser
Mitsuo Motoki
Shay Mozes
Kamesh Munagala
Nikos Mutsanas
Georgios Mylonas
Hiroshi Nagamochi
Danupon Nanongkai
Seffi Naor
Alfredo Navarra
Ilan Newman
Gaia Nicosia
Martin Niemeier
Sotiris Nikoletseas
Evdokia Nikolova
Harumichi Nishimura
Gabriel Nivasch
Tim Nonner
Adrian Ogierman
Mitsunori Ogihara
Yoshio Okamoto
Krzysztof Onak
Hirotaka Ono
Christina Otte
Sang-il Oum
Konstantinos Panagiotou
Panagiota Panagopoulou
Vangelis Paschos
Mihai Patrascu
Arno Pauly
Andreas Pavlogiannis
Rudi Pendavingh
Xavier Perez-Gimenez
Ludovic Perret
Giuseppe Persiano
Seth Pettie
Ulrich Pferschy
George Pierrakos
Peter Pietrzyk
CK Poon
Ely Porat
Lars Prädel
Guido Proietti
Kirk Pruhs
Apostolos Pyrgelis
Maurice Queyranne
Charles Rackoff
Harald Raecke
Prasad Raghavendra
Sagunthevar Rajasekaran
Rajeev Raman
Suneeta Ramaswami
Dominik Raub
Bala Ravikumar
Dror Rawitz
Igor Razgon
Andrea Ribichini
Stephane Robert
Liam Roditty
Heiko Roeglin
Alon Rosen
Thomas Rothvoss
Atri Rudra
Amin Saberi
Amit Sahai
Roy Sambuddha
Piotr Sankowski
Rahul Santhanam
Tamas Sarlos
Saket Saurabh
Christian Schaffner
Peter Scheiblechner
Ingo Schiermeyer
Stefan Schmid
Grant Schoenebeck
Uwe Schoening
Florian Schoppmann

Ulf-Peter Schroeder
Warren Schudy
Andreas S. Schulz
Ulrich M. Schwarz
Nathan Segerlind
Gil Segev
Raimund Seidel
Siddhartha Sen
Rocco Servedio
Hadas Shachnai
Ronen Shaltiel
Yaoyun Shi
Amir Shpilka
Riccardo Silvestri
Alistair Sinclair
Yaron Singer
Rene Sitters
Martin Skutella
Michiel Smid
Anthony Man-Cho So
Christian Sohler
Roberto Solis-Oba
Reto Spöhel
Ioannis Stamatiou
David Steurer
Sebastian Stiller
Miloš Stojaković
James Storer
Martin Strauss
Marek Sulovský
Subhash Suri
Ola Svensson
Mohammad Taghi Hajiaghayi
Suguru Tamaki
Chee Wei Tan
Keisuke Tanaka
Kanat Tangwongsan
Seiichiro Tani
Evimaria Terzi
Stefano Tessaro
Henning Thomas
Marc Thurley
Hingfung Ting
Denis Trystram
Tobias Tscheuschner
Kei Uchizawa
Kenya Ueno
Marc Uetz
Walter Unger
Salil Vadhan
Yevgeniy Vahlis
Vinod Vaikuntanathan
Marc van Barel
Rob van Stee
Stefan van Zwam
Harsha Vardhan Simhadri
Virginia Vassilevska Williams
Sergei Vassilvitskii
Umesh V. Vazirani
Dan Vilenchik
Ivan Visconti
Joachim von zur Gathen
Tjark Vredeveld
Fabian Wagner
Uli Wagner
John Watrous
Oren Weimann
Matthias Westermann
Peter Widmayer
Ryan Williams
Philipp Woelfel
Duncan Wong
Jurg Wullschlege
Masaki Yamamoto
Shigeru Yamashita
Guomin Yang
Deshi Ye
Sergey Yekhanin
Hsu-Chun Yen
Ke Yi
Ming Yin
Neal Young
Raphael Yuster
Wenan Zang
Christos Zaroliagis
Guochuan Zhang
Jie Zhang
Qiang Zhang
Zhenjie Zhang
Yunlei Zhao
Hong-Sheng Zhou
Uri Zwick

Table of Contents – Part I

Invited Talks

Local Search: Simple, Successful, But Sometimes Sluggish 1
Burkhard Monien, Dominic Dumrauf, and Tobias Tscheuschner

When Conflicting Constraints Can be Resolved – The Lovász Local Lemma and Satisfiability (Abstract) 18
Emo Welzl

Session 1-Track A. Combinatorial Optimization

Plane Spanners of Maximum Degree Six 19
Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perković

The Positive Semidefinite Grothendieck Problem with Rank Constraint 31
Jop Briët, Fernando Mário de Oliveira Filho, and Frank Vallentin

Cycle Detection and Correction 43
Amihood Amir, Estrella Eisenberg, Avivit Levy, Ely Porat, and Natalie Shapira

Decomposition Width of Matroids 55
Daniel Král'

Session 2-Track A1. Game Theory

The Cooperative Game Theory Foundations of Network Bargaining Games 67
MohammadHossein Bateni, MohammadTaghi Hajiaghayi, Nicole Immorlica, and Hamid Mahini

On the Existence of Pure Nash Equilibria in Weighted Congestion Games 79
Tobias Harks and Max Klimm

On the Limitations of Greedy Mechanism Design for Truthful Combinatorial Auctions 90
Allan Borodin and Brendan Lucier

Mean-Payoff Games and Propositional Proofs 102
Albert Atserias and Elitza Maneva

Session 2-Track A2. Security

Online Network Design with Outliers 114
Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Piotr Sankowski

Efficient Completely Non-malleable Public Key Encryption 127
Benoît Libert and Moti Yung

Polynomial-Space Approximation of No-Signaling Provers 140
Tsuyoshi Ito

From Secrecy to Soundness: Efficient Verification via Secure Computation (Extended Abstract) 152
Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz

Session 3-Track A1. Data Structures

Mergeable Dictionaries 164
John Iacono and Özgür Özkan

Faster Algorithms for Semi-matching Problems (Extended Abstract) . . . 176
Jittat Fakcharoenphol, Bundit Laekhanukit, and Danupon Nanongkai

Clustering with Diversity 188
Jian Li, Ke Yi, and Qin Zhang

New Data Structures for Subgraph Connectivity 201
Ran Duan

Session 3-Track A2. Sorting & Hashing

Tight Thresholds for Cuckoo Hashing via XORSAT (Extended Abstract) 213
Martin Dietzfelbinger, Andreas Goerdt, Michael Mitzenmacher, Andrea Montanari, Rasmus Pagh, and Michael Rink

Resource Oblivious Sorting on Multicores 226
Richard Cole and Vijaya Ramachandran

Interval Sorting 238
Rosa M. Jiménez and Conrado Martínez

Session 4-Track A. Graphs, Nets and Optimization

Inapproximability of Hypergraph Vertex Cover and Applications to Scheduling Problems 250
Nikhil Bansal and Subhash Khot

Thresholded Covering Algorithms for Robust and Max-min Optimization 262
Anupam Gupta, Viswanath Nagarajan, and R. Ravi

Graph Homomorphisms with Complex Values: A Dichotomy Theorem (Extended Abstract) 275
Jin-Yi Cai, Xi Chen, and Pinyan Lu

Metrical Task Systems and the k-Server Problem on HSTs 287
Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor

Session 5-Track A1. Scheduling

Scheduling Periodic Tasks in a Hard Real-Time Environment 299
Friedrich Eisenbrand, Nicolai Hähnle, Martin Niemeier, Martin Skutella, José Verschae, and Andreas Wiese

Scalably Scheduling Power-Heterogeneous Processors 312
Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs

Better Scalable Algorithms for Broadcast Scheduling 324
Nikhil Bansal, Ravishankar Krishnaswamy, and Viswanath Nagarajan

Max-min Online Allocations with a Reordering Buffer 336
Leah Epstein, Asaf Levin, and Rob van Stee

Session 5-Track A2. Graphs & Hypergraphs

Orientability of Random Hypergraphs and the Power of Multiple Choices 348
Nikolaos Fountoulakis and Konstantinos Panagiotou

On the Inapproximability of Vertex Cover on k-Partite k-Uniform Hypergraphs 360
Venkatesan Guruswami and Rishi Saket

Dynamic Programming for Graphs on Surfaces 372
Juanjo Rué, Ignasi Sau, and Dimitrios M. Thilikos

Interval Graphs: Canonical Representation in Logspace 384
Johannes Köbler, Sebastian Kuhnert, Bastian Laubner, and Oleg Verbitsky

Session 6-Track A. Best Paper Award

Approximating the Partition Function of the Ferromagnetic Potts Model 396
Leslie Ann Goldberg and Mark Jerrum

Session 7-Track A. Algebraic Problems

On the Relation between Polynomial Identity Testing and Finding Variable Disjoint Factors 408
Amir Shpilka and Ilya Volkovich

On Sums of Roots of Unity 420
Bruce Litow

Exponential Time Complexity of the Permanent and the Tutte Polynomial (Extended Abstract) 426
Holger Dell, Thore Husfeldt, and Martin Wahlén

On Approximate Horn Formula Minimization 438
Amitava Bhattacharya, Bhaskar DasGupta, Dhruv Mubayi, and György Turán

Session 8-Track A. Networks & Communication Complexity

Choosing, Agreeing, and Eliminating in Communication Complexity 451
Amos Beimel, Sebastian Ben Daniel, Eyal Kushilevitz, and Enav Weinreb

Additive Spanners in Nearly Quadratic Time 463
David P. Woodruff

Composition Theorems in Communication Complexity 475
Troy Lee and Shengyu Zhang

Network Design via Core Detouring for Problems without a Core 490
Fabrizio Grandoni and Thomas Rothvoß

Session 9-Track A1. Complexity & Automata

Weak Completeness Notions for Exponential Time 503
Klaus Ambos-Spies and Timur Bakibayev

Efficient Evaluation of Nondeterministic Automata Using Factorization Forests 515
Mikołaj Bojańczyk and Paweł Parys

On the Complexity of Searching in Trees: Average-Case Minimization 527
Tobias Jacobs, Ferdinando Cicalese, Eduardo Laber, and Marco Molinaro

Session 9-Track A2. Finding & Testing

Finding Is as Easy as Detecting for Quantum Walks 540
Hari Krovi, Frédéric Magniez, Maris Ozols, and Jérémie Roland

Improved Constructions for Non-adaptive Threshold Group Testing 552
Mahdi Cheraghchi

Testing Non-uniform k-Wise Independent Distributions over Product Spaces (Extended Abstract) 565
Ronitt Rubinfeld and Ning Xie

Session 10-Track A1. Approximations

A Sublogarithmic Approximation for Highway and Tollbooth Pricing . . . 582
Iftah Gamzu and Danny Segev

Maximum Quadratic Assignment Problem: Reduction from Maximum Label Cover and LP-Based Approximation Algorithm 594
Konstantin Makarychev, Rajsekar Manokaran, and Maxim Sviridenko

Cell Probe Lower Bounds and Approximations for Range Mode 605
Mark Greve, Allan Grønlund Jørgensen, Kasper Dalgaard Larsen, and Jakob Truelsen

SDP Gaps for 2-to-1 and Other Label-Cover Variants 617
Venkatesan Guruswami, Subhash Khot, Ryan O'Donnell, Preyas Popat, Madhur Tulsiani, and Yi Wu

Session 10-Track A2. Streaming & Preprocessing

Data Stream Algorithms for Codeword Testing (Extended Abstract) 629
Atri Rudra and Steve Uurtamo

Streaming Algorithms for Independent Sets 641
Bjarni V. Halldórsson, Magnús M. Halldórsson, Elena Losievskaja, and Mario Szegedy

Preprocessing of Min Ones Problems: A Dichotomy 653
Stefan Kratsch and Magnus Wahlström

Holographic Reduction: A Domain Changed Application and Its Partial Converse Theorems 666
Mingji Xia

Session 11-Track A1. Adaptive, Knowledge & Optimality

Optimal Trade-Offs for Succinct String Indexes 678
Roberto Grossi, Alessio Orlandi, and Rajeev Raman

Approximation Algorithms for Optimal Decision Trees and Adaptive TSP Problems 690
Anupam Gupta, Viswanath Nagarajan, and R. Ravi

Concurrent Knowledge Extraction in the Public-Key Model 702
Andrew C. Yao, Moti Yung, and Yunlei Zhao

Session 11-Track A2. Covering, Graphs & Independence

On the k-Independence Required by Linear Probing and Minwise Independence 715
Mihai Pătraşcu and Mikkel Thorup

Covering and Packing in Linear Space 727
Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto

Testing 2-Vertex Connectivity and Computing Pairs of Vertex-Disjoint s-t Paths in Digraphs 738
Loukas Georgiadis

Author Index 751

Table of Contents – Part II

Invited Talks

Informative Labeling Schemes (Abstract) 1
Pierre Fraigniaud

Noetherian Spaces in Verification 2
Jean Goubault-Larrecq

Towards a Theory of Time-Bounded Verification 22
Joël Ouaknine and James Worrell

Physical Algorithms 38
Roger Wattenhofer

Session 1-Track B. Automata

Optimal Zielonka-Type Construction of Deterministic Asynchronous Automata 52
Blaise Genest, Hugo Gimbert, Anca Muscholl, and Igor Walukiewicz

Pumping and Counting on the Regular Post Embedding Problem 64
Pierre Chambart and Philippe Schnoebelen

Alternation Removal in Büchi Automata 76
Udi Boker, Orna Kupferman, and Adin Rosenberg

Linear Orders in the Pushdown Hierarchy 88
Laurent Braud and Arnaud Carayol

Session 1-Track C. Communication in Networks

The Serializability of Network Codes 100
Anna Blasiak and Robert Kleinberg

How Efficient Can Gossip Be? (On the Cost of Resilient Information Exchange) 115
Dan Alistarh, Seth Gilbert, Rachid Guerraoui, and Morteza Zadimoghaddam

Efficient Information Exchange in the Random Phone-Call Model 127
Petra Berenbrink, Jurek Czyzowicz, Robert Elsässer, and Leszek Gąsieniec

An $O(\log n)$-Competitive Online Centralized Randomized Packet-Routing Algorithm for Lines 139
Guy Even and Moti Medina

Session 2-Track B. Formal Languages

A Topological Approach to Recognition 151
Mai Gehrke, Serge Grigorieff, and Jean-Éric Pin

On $LR(k)$-Parsers of polynomial Size (Extended Abstract) 163
Norbert Blum

On Erasing Productions in Random Context Grammars 175
Georg Zetzsche

Session 4-Track B. Semantics

Game Semantics for Call-by-Value Polymorphism 187
James Laird

What is a Pure Functional? 199
Martin Hofmann, Aleksandr Karbyshev, and Helmut Seidl

Example-Guided Abstraction Simplification 211
Roberto Giacobazzi and Francesco Ranzato

Compositional Closure for Bayes Risk in Probabilistic Noninterference 223
Annabelle McIver, Larissa Meinicke, and Carroll Morgan

Session 4-Track C. Fault Tolerance, Ranking

Asynchronous Throughput-Optimal Routing In Malicious Networks 236
Paul Bunn and Rafail Ostrovsky

Improved Fault Tolerance and Secure Computation on Sparse Networks 249
Nishanth Chandran, Juan Garay, and Rafail Ostrovsky

Sparse Reliable Graph Backbones 261
Shiri Chechik, Yuval Emek, Boaz Patt-Shamir, and David Peleg

Approximation Algorithms for Diversified Search Ranking 273
Nikhil Bansal, Kamal Jain, Anna Kazeykina, and Joseph (Seffi) Naor

Session 5-Track B. Graphs, Categories and Quantum Information

Rewriting Measurement-Based Quantum Computations with Generalised Flow 285
Ross Duncan and Simon Perdrix

The Compositional Structure of Multipartite Quantum Entanglement . . . 297
Bob Coecke and Aleks Kissinger

Compositionality in Graph Transformation 309
Arend Rensink

Session 6-Track B. Best Paper Award

On p-Optimal Proof Systems and Logics for PTIME 321
Yijia Chen and Jörg Flum

Session 6-Track C. Best Paper Award

Placing Regenerators in Optical Networks to Satisfy Multiple Sets of Requests 333
George B. Mertzios, Ignasi Sau, Mordechai Shalom, and Shmuel Zaks

Session 7-Track B. Logic

Maximal Decidable Fragments of Halpern and Shoham's Modal Logic of Intervals 345
Angelo Montanari, Gabriele Puppis, and Pietro Sala

B and D Are Enough to Make the Halpern–Shoham Logic Undecidable 357
Jerzy Marcinkowski, Jakub Michaliszyn, and Emanuel Kieroński

Parameterized Modal Satisfiability 369
Antonis Achilleos, Michael Lampis, and Valia Mitsou

Automata for Coalgebras: An Approach Using Predicate Liftings 381
Gaëlle Fontaine, Raul Leal, and Yde Venema

Session 7-Track C. Privacy, Selfishness

Resolving the Complexity of Some Data Privacy Problems 393
Jeremiah Blocki and Ryan Williams

Private and Continual Release of Statistics 405
T-H. Hubert Chan, Elaine Shi, and Dawn Song

Envy-Free Pricing in Multi-item Markets 418
Ning Chen and Xiaotie Deng

Contention Resolution under Selfishness 430
George Christodoulou, Katrina Ligett, and Evangelia Pyrga

Session 8-Track B. Concurrency

On the Expressiveness of Polyadic and Synchronous Communication in Higher-Order Process Calculi 442
Ivan Lanese, Jorge A. Pérez, Davide Sangiorgi, and Alan Schmitt

On Bisimilarity and Substitution in Presence of Replication 454
Daniel Hirschkoff and Damien Pous

The Downward-Closure of Petri Net Languages 466
Peter Habermehl, Roland Meyer, and Harro Wimmel

Reachability Games on Extended Vector Addition Systems with States 478
Tomáš Brázdil, Petr Jančar, and Antonín Kučera

Session 8-Track C. Mobile Agents

Modelling Mobility: A *Discrete* Revolution (Extended Abstract) 490
Andrea E.F. Clementi, Angelo Monti, and Riccardo Silvestri

Tell Me Where I Am So I Can Meet You Sooner: (Asynchronous Rendezvous with Location Information) 502
Andrew Collins, Jurek Czyzowicz, Leszek Gąsieniec, and Arnaud Labourel

Rendezvous of Mobile Agents without Agreement on Local Orientation 515
Jérémie Chalopin and Shantanu Das

Session 9-Track B. Probabilistic Computation

Probabilistic Automata on Finite Words: Decidable and Undecidable Problems 527
Hugo Gimbert and Youssouf Oualhadj

Space-Efficient Scheduling of Stochastically Generated Tasks 539
Tomáš Brázdil, Javier Esparza, Stefan Kiefer, and Michael Luttenberger

Exponential Lower Bounds For Policy Iteration 551
John Fearnley

Session 10-Track B. Automata

Regular Temporal Cost Functions 563
Thomas Colcombet, Denis Kuperberg, and Sylvain Lombardy

Model Checking Succinct and Parametric One-Counter Automata 575
Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell

Pebble Weighted Automata and Transitive Closure Logics 587
Benedikt Bollig, Paul Gastin, Benjamin Monmege, and Marc Zeitoun

Energy Parity Games 599
Krishnendu Chatterjee and Laurent Doyen

Author Index 611

Local Search: Simple, Successful, But Sometimes Sluggish*

Burkhard Monien, Dominic Dumrauf, and Tobias Tscheuschner

Faculty of Computer Science, Electrical Engineering and Mathematics,
University of Paderborn, Fürstenallee 11, 33102 Paderborn, Germany
{bm,dumrauf,chessy}@uni-paderborn.de

Abstract. In this paper, we survey the research on the complexity of computing locally optimal solutions. We revisit well-known and successful local search heuristics and present results on the complexity of the frequently used standard local search algorithm for various problems. Here, our focus in on worst case, average case, and smoothed complexity along with the existence of sequences of improving steps of exponential length. For a more theoretical investigation, we revisit the framework of $\mathcal{PLS}$ and mostly concentrate on the research on CongestionGames, which sparked the interconnection between local search and game theory. We conclude by stating various open problems.

1 Introduction

Optimization problems occur in many areas of everyday life. If we are at point A and want to reach point B, we try to minimize the length of the path connecting A and B. If we decide between several leisure time options – a football game, visiting the theater or going to the casino – we try to maximize our personal utility. When encountered by an optimization problem, we are mostly interested in two measures: the quality of a solution and the time needed to find it. If the utility that we expect by finding a (better) solution is greater than the expected cost due to the time required to find it, then it is beneficial to search for a (better) solution. In this respect, the estimation of the required time plays a crucial role in the process of optimization.

For many optimization problems, it is known to be $\mathcal{NP}$-complete to find a global optimum. Since no polynomial time algorithm is known that computes global optima for such problems, several approaches were developed to find at least *good* solutions. For instance, approximation algorithms compute solutions that have an objective function value which is not more than a predetermined factor away from an optimum. Unfortunately, some problems in $\mathcal{NP}$ even resist polynomial time approximation in the sense that the existence of a polynomial time algorithm that computes an approximate solution would directly imply $\mathcal{P} = \mathcal{NP}$. A popular approach to tackle those problems are (meta-)heuristics.

* This work is partially supported by German Research Foundation (DFG) Priority Program 1307 Algorithm Engineering.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 1–17, 2010.

Nearly all metaheuristics – local search, simulated annealing, evolutionary algorithms, to name a few popular ones – compute from a given (population of) solution(s) a new (population of) solution(s) and continue the computation with the new one(s), whereby solutions with a higher quality with respect to the objective function are preferred. In case of the local search approach the preference is strict, i.e. the computation is only continued with strictly better solutions. For convex optimization problems, this approach is most suitable since local optima coincide with global optima in convex optimization problems. The advantage of metaheuristics with non-strict preference for new solutions is that the computation can escape local optima. This is mostly of use if local optima are frequent in the solution space but of rather different quality with respect to the objective function. However, the focus of this survey is the local search approach and in particular the complexity of computing a local optimum.

Local search is one of the most frequently used approach to solve hard combinatorial optimization problems. Famous examples of successful application of local search algorithms are the simplex method for solving linear programs, [9,64], the k-opt heuristic for finding solutions of the Traveling Salesman Problem, [2], and the k-means algorithm for clustering objects, [18,30]. For further information on local search, its complexity, and related problems we refer the reader to [62,71,72].

Roadmap. The paper is organized as follows: Section 2 considers three combinatorial optimization problems in which local search was most successfully applied and discusses their complexity. Section 3 presents the class $\mathcal{PLS}$, the concept of tight reductions and its implications. We focus on $\mathcal{PLS}$-complete problems in Section 4, where we first survey the early works on fundamental combinatorial optimization problems such as Satisfiability and the Traveling Salesman Problem and then proceed by presenting an overview of recent results on CongestionGames in the area of game theory. We conclude the paper by outlining open problems.

2 Successful Applications of Local Search

In this section we consider three combinatorial optimization problems in which local search celebrated its greatest practical successes, namely linear programs, the traveling salesman problem, and the problem of clustering objects.

LinearPrograms. In a linear program the input is a matrix A and vectors b, c and the task is to find a vector x maximizing $c^T x$ such that $Ax \leq b$. Due to the convexity of linear programs, local optima coincide with global optima which emphasizes the use of local search in a natural way. In 1947, George Danzig, [16], introduced the famous Simplex Algorithm which finds an optimum by starting at a vertex of the polytope formed by the constraints $Ax \leq b$ and iteratively hopping to better vertices with respect to $c^T x$ until an optimum is reached. Since that time, the Simplex Algorithm was successfully applied to linear programs

originating from a wide range of applications including scheduling problems, production planning, routing problems, and game theory. Although the running time of the simplex algorithm was already observed to be polynomial on "real-world" instances, significant progress in speeding up the algorithm was achieved since the end of the 1980s, [9].

In contrast to the low running time observed for practical instances the existence of exponentially long improving sequences raised interest in the early years of the simplex method. For the steepest descent pivoting rule, linear programs were constructed for which the simplex method takes an exponential number of pivoting steps, [67]. For other pivot rules similar results were shown, [65,64]. On the other hand, Kalai and Kleitman, [33], showed that for every initial solution of a linear program with n inequalities and d variables the simplex algorithm is at most $n^{\log d+2}$ pivot steps away from a local optimum. This raises the question whether it is possible to find this path efficiently. However, computing an optimum point of a linear program is known to be in $\mathcal{P}$ since Khachiyan, [35], introduced his Ellipsoid method which uses an approach different from local search. Karmarker, [34], subsequently introduced the interior point method which also needs polynomial time and even outperforms the simplex algorithm in some practical applications.

Traveling Salesman Problem. In the TSP the input is an undirected weighted complete graph and the output is a cycle of minimum weight that visits all nodes of the graph. The probably most frequently used local search heuristic for this problem is 2-OPT. It starts with an initial tour and iteratively improves it by exchanging two edges of the tour by two different ones as long as such an improving step is possible. For random and "real-world" Euclidean instances this heuristic is known to compute very good tours within a sub-quadratic number of improving steps, [31,55].

On the other hand, it was shown that there are instances and initial solutions of TSP for which the k-opt heuristic for $k \geq 2$ can take exponentially many improving steps, [12,43]. However, these instances did not fulfill the triangle inequality and the question whether such instances can be constructed for the metric TSP remained open for a long time. Finally, Englert et al. [21] found Euclidean instances for which the 2-opt heuristic can take exponentially many improving steps.

Clustering. The problem of clustering asks for a partition of a set of data into subsets (the clusters) such that some given measure for the similarity within the clusters is maximized. The problem of clustering occurs in many applications including pattern recognition, data compression, and load balancing and depending on the application in different forms. A well studied algorithm for clustering points in the Euclidean space is the k-means algorithm. It starts with an initial set of k centers for the clusters, whereby each data points is assigned to the closest center. Then, it improves the solution by repeatedly performing the following two steps. At first, for each cluster a new center is defined as the average of all points of the cluster, i.e. the "mean", and then all points are

assigned to the cluster represented by the closest of the new center points. Note that in each improving step the sum of the distances of the data points to their corresponding closest center, which can be treated as a potential function, decreases. We remark that in the Euclidean space such an improving step of the k-means algorithm is uniquely determined.

For the k-means algorithm the number of steps was observed to be linear in the number of data points on practical instances, [18]. And similarly as for the two previously mentioned famous problems there are instances and initial solutions of the clustering problem for which the k-means algorithm takes an exponential number of improving steps to converge, [66].

2.1 Randomized Instances and Smoothed Complexity

For many years, it was observed that the running time of local search algorithms, and in particular the simplex method, on most instances occurring in practical applications was very low. Inspired by this observation, the complexity of the simplex algorithm was considered for many distributions of random inputs and shown to be in expected polynomial time, [5,10,61]. The same observation was made for the 2-opt heuristic for computing solutions of the TSP on random instances in the unit hypercube, [12]. However, as for the constructed inputs for which an exponential number of improving steps are possible, it can be argued that the random instances may have certain properties that do not reflect the properties of real-world instances.

To understand why the running time is polynomial on so many real-world instances, Spielmann and Teng, [63], introduced the notion of smoothed complexity which measures the expected running time of an algorithm under small random perturbations of the input. They showed that the simplex algorithm for linear programs has polynomial smoothed complexity. Subsequently, the notion of smoothed complexity was adapted for algorithms of various other problems including other local search algorithms. The smoothed complexity of the 2-opt heuristic for Euclidean instances was shown to be polynomial, [21]. And in a recent paper the k-means algorithm was also shown to have polynomial smoothed complexity, [7].

3 $\mathcal{PLS}$, Tight Reductions, and Completeness

The fundamental definitions of a $\mathcal{PLS}$-problem, the class $\mathcal{PLS}$, and tight $\mathcal{PLS}$-reductions were introduced by Johnson, Papadimitriou, Schäffer, and Yannakakis, [32,49,57]. In the following definitions, we assume that all elements of all occurring sets are encoded as binary strings of finite length.

A local search problem Π consists of a set of instances $\mathcal{I}$, a set of feasible solutions $F_\Pi(I)$ for every instance $I \in \mathcal{I}$, where the length of every solution is bounded by a polynomial in the length of I, and an objective function $f : F_\Pi(I) \to \mathbb{Z}$. In addition, every solution $\mathsf{s} \in F_\Pi(I)$ has a neighborhood

$N_\Pi(\mathsf{s}, I) \subseteq F_\Pi(I)$. For an instance $I \in \mathcal{I}$, the problem is to find a solution $\mathsf{s} \in F_\Pi(I)$ such that for all $\mathsf{s}' \in N_\Pi(\mathsf{s}, I)$ solution s' does not have a greater value than s with respect to f in case of maximization and not a lower value in case of minimization.

A local search problem Π is in the class $\mathcal{PLS}$, [32], if the following three polynomial time algorithms exist: algorithm $\textsc{Init}_\Pi(I)$ computes for every instance $I \in \mathcal{I}$ a feasible solution $\mathsf{s} \in F_\Pi(I)$, algorithm $\textsc{Cost}_\Pi(I, \mathsf{s})$ computes for every $I \in \mathcal{I}$ and $\mathsf{s} \in F_\Pi(I)$ the value $f(\mathsf{s})$, and algorithm $\textsc{Improve}_\Pi(\mathsf{s}, I)$ returns for every $I \in \mathcal{I}$ and $\mathsf{s} \in F_\Pi(I)$ a better neighbor solution $\mathsf{s}' \in N_\Pi(\mathsf{s}, I)$ if there is one and "locally optimal" otherwise.

Locally optimal solutions can be found non-deterministically in polynomial time by first guessing a feasible solution and subsequently verifying local optimality. It is known that if a $\mathcal{PLS}$-problem L is $\mathcal{NP}$-hard, then $\mathcal{NP}$ is closed under complement, [32].

Consider the following simple minimization $\mathcal{PLS}$-problem Exp: Instances are natural numbers $n \in \mathbb{N}$, encoded as a binary string. The set of feasible solutions consists of all binary strings $\{0,1\}^n$. The cost of a solution is the natural number the binary string represents and the single neighbor is the natural number the binary string represents minus one; the neighborhood of the all-zero vector is empty. Starting with the all-one vector, it is obvious to see that every sequence of improving steps requires exponentially many steps to reach the locally optimal all-zero vector.

Definition 1. *We say that a $\mathcal{PLS}$-problem has the* all-exp property *if there exists an instance I and an initial feasible solution for I that is exponentially many improving steps away from any local optimum. A sequence has* exponential length *if its number of improving steps is at least $2^{\sqrt[k]{|I|}}$ for some $k \in \mathbb{N}$. A $\mathcal{PLS}$-problem has the* is-exp property *if there exist instances such that there is a sequence of improving steps of exponential length.*

Note that by definition, the class all-exp is closed under polynomial reductions. Furthermore, problems for which each solution has at most one better neighbor solution the is-exp property directly implies the all-exp property, in particular this is the case for the problem Exp and the clustering problem with neighborhood superimposed by the k-means algorithm. Remarkably, for the combinatorial optimization problems where local search was applied in practice as outlined in Section 2 the is-exp property was shown but not the all-exp property.

The *standard algorithm problem* is, for given instance I of $\mathcal{PLS}$-problem L and some feasible solution $\mathsf{s} \in F_L(I)$, to compute a local optimum $\mathsf{s}^\star \in F_L(I)$ which is reachable from s by successive improvements. There exists a $\mathcal{PLS}$-problem whose standard algorithm problem is $\mathcal{PSPACE}$-complete, [49,70].

A problem $\Pi \in \mathcal{PLS}$ is *$\mathcal{PLS}$-reducible* to another problem $\Pi' \in \mathcal{PLS}$ (written $\Pi \leq_{\text{pls}} \Pi'$) if the following polynomial time computable functions Φ and Ψ exist. The function Φ maps instances I of Π to instances of Π' and Ψ maps pairs (s, I), where s is a solution of $\Phi(I)$, to solutions of I, such that for all instances

I of Π and local optima $\mathsf{s}^\star$ of $\Phi(I)$ the solution $\Psi(\mathsf{s}^\star, I)$ is a local optimum of I. Finally, a problem $\Pi \in \mathcal{PLS}$ is *$\mathcal{PLS}$-complete* if every problem in $\mathcal{PLS}$ is $\mathcal{PLS}$-reducible to Π. In a nutshell, a $\mathcal{PLS}$-reduction is *tight* if all sequences of improving steps in the reduced instance correspond to sequences of improving steps in the original problem whose length may only be increased by introducing intermediate solutions. For a formal definition, confer Papadimitriou et al., [49].

The concept of $\mathcal{PLS}$-completeness is in line with the general use of completeness in complexity theory. In this case, the concept entails that if *some* $\mathcal{PLS}$-complete problem is polynomial-time solvable, then *all* problems in the class $\mathcal{PLS}$ are polynomial-time solvable. Tight reductions are of special interest, since they preserve the $\mathcal{PSPACE}$-completeness of the standard algorithm problem, [49,70], as well as the all-exp property, [32]. Note that tight reductions are transitive. This allows to define the tight $\mathcal{PLS}$-completess of $\mathcal{PLS}$-problems recursively: $\mathcal{PLS}$-problem Circuit/Flip is tight $\mathcal{PLS}$-complete—we will justify this in Section 4.1. A $\mathcal{PLS}$-problem B is *tight $\mathcal{PLS}$-complete* if there exists a tight $\mathcal{PLS}$-reduction $A \leq_{\text{pls}} B$ for some tight $\mathcal{PLS}$-complete problem A. Let us remark that linear programming with the simplex neighborhood does not have the all-exp property, [33], and is therefore not tight $\mathcal{PLS}$-complete.

4 $\mathcal{PLS}$-Complete Problems

In this section, we first focus on results for the fundamental problem Circuit/Flip. We continue by presenting results for the Traveling Salesman Problem, MaxCut, and Satisfiability. Most of these pioneering results are from Johnson, Papadimitriou, and Yannakakis, [32,49], or from Schäffer and Yannakakis, [57,70]. We close by surveying recent results from game theory considering the complexity of computing pure Nash equilibria. For further known $\mathcal{PLS}$-complete problems, we refer the reader to [6,11,36,37,49,53,59,69]. For a general overview, confer the books of Aarts et al., [1], and Aarts and Lenstra, [2], of which the first one contains a list of $\mathcal{PLS}$-complete problems known so far. Unless otherwise mentioned, for the remainder of this section, we assume that all numbers are integers.

4.1 Early Results

The early results in the field of $\mathcal{PLS}$ were motivated by developing a theoretical framework to investigate the successful local search algorithms presented in Section 2. The main focus was on the complexity of computing a locally optimal solution for well-known hard combinatorial optimization problems.

Circuit/Flip. The first and generic $\mathcal{PLS}$-problem which was proven to be $\mathcal{PLS}$-complete is Circuit/Flip, [32]. In the Circuit/Flip problem, the task is to find a binary input vector for a given feedback-free boolean circuit $\mathcal{S}$ such that the output vector, treated as a binary number, cannot be increased by

flipping a single input bit. The hardness proof given by Johnson, Papadimitriou, and Yannakakis, [32], involves three intermediate reductions: First, they reduce an arbitrary given problem $L \in \mathcal{PLS}$ to an intermediate problem L' which only differs from L by each solution $\mathsf{s} \in F_L(I)$ (a binary string of length $p(|I|)$) in each instance $I \in L$ having at most one neighbor. For this, the single neighbor of s is defined to be the output of $\textsc{Improve}_L(\mathsf{s}, I)$; call the resulting instance I' and the problem L'. Secondly, they reduce to a $\mathcal{PLS}$-problem L'' which has the set of solutions of L', but every bit string of length $p(|I|)$ is now a feasible solution and two solutions are mutual neighbors if they differ in a single bit. Note that this exactly matches the neighborhood structure of Circuit/Flip. The cost function is defined such that every minimum length sequence of bitflips between two solutions $\mathsf{s}, \mathsf{t} \in F_{L'}(I')$ yields an improving path $\mathsf{s} \rightsquigarrow \mathsf{t}$ if and only if $\mathsf{t} = N_{L'}(\mathsf{s}, I')$. The crucial idea here is to assign each intermediate solution u on each such minimum length sequence $\mathsf{s} \rightsquigarrow \mathsf{t}$ the cost of t scaled by some factor plus the Hamming distance between u and t. The final reduction to Circuit/Flip simply involves constructing a feedback-free boolean circuit which computes the cost function of I''. For all technical details, we refer the reader to the original paper, [32].

Notably, this reduction is tight, [49]. Hence, exponentially long sequences of improving steps from initial feasible solutions in $\mathcal{PLS}$-problem Exp are preserved by the reduction and subsequently Circuit/Flip possesses the all-exp property. Furthermore, the tightness of the reduction implies that the standard algorithm problem for Circuit/Flip is $\mathcal{PSPACE}$-complete, since the $\mathcal{PLS}$-problem whose standard algorithm problem is $\mathcal{PSPACE}$-complete can be directly simulated using Circuit/Flip. By definition, Circuit/Flip is tight $\mathcal{PLS}$-complete. Hence, problems shown to be tight $\mathcal{PLS}$-complete via a sequence of tight reductions from Circuit/Flip are again tight $\mathcal{PLS}$-complete, implying these problems possess the all-exp property and their standard algorithm problem is $\mathcal{PSPACE}$-complete.

Traveling Salesman Problem. As outlined in Section 2, local search algorithms have been applied with huge success to compute approximate solutions for the Traveling Salesman Problem on random Euclidean instances, [31]. On the negative side, computing a locally optimal solution with respect to the Lin-Kernighan heuristic (confer Johnson and McGeoch, [31], for an exact definition), [42], is $\mathcal{PLS}$-complete, [49,50], such as finding a locally optimal solution for the Traveling Salesman Problem where the neighborhood structure is superimposed by the well-known k-OPT heuristic, [41], for some $k \gg 1,000$, [38]. In the k-OPT neighborhood, two solutions are mutual neighbors if they differ in at most k edges.

MaxCut. For an undirected graph $G = (V, E)$ with weighted edges $w : E \to \mathbb{N}$, a cut is a partition of V into two sets V_1, V_2. The weight of the cut is the sum of the weights of the edges connecting nodes between V_1 and V_2. The MaxCut-problem asks for a cut of maximum weight and the problem is known to be tight $\mathcal{PLS}$-complete, [57]. Let us remark that in the reduction, the maximum degree

of the nodes in the graph is required to be unbounded. If the maximum degree in the input graph is at most three, then there are at most quadratically many improving steps, [52]. Hence, local optima can be computed in polynomial time via successive improvements. This does no longer hold for graphs with maximum degree four, as it was recently shown that MaxCut has the all-exp property, even on graphs with maximum degree four, [47].

Satisfiability. In their seminal paper, Johnson, Papadimitriou, and Yannakakis conjecture that for a problem to be $\mathcal{PLS}$-complete, the problem of verifying local optimality is required to be $\mathcal{P}$-complete. Krentel, [39], disproves this conjecture by showing that a weighted natural local search version of Satisfiability (confer problem [L01] in Garey and Johnson, [27], for a formal definition) is $\mathcal{PLS}$-complete, though the problem of verifying local optimality can be solved in Logspace. For problems similar to Satisfiability, stronger results can be shown. The Maximum Constraint Assignment-problem is a natural local search version of weighted, GeneralizedSatisfiability (confer [L06] in Garey and Johnson, [27] for a formal definition), extended to higher valued variables in constraints. Here, constraints map assignments for variables to integers. Additional parameters in (p, q, r)-MCA simultaneously limit the maximum number of variables p each constraint depends upon, the maximum appearance q of each variable and its valuedness r. The problem (4,3,3)-MCA is known to be $\mathcal{PLS}$-complete and this result can be extended to Satisfiability over binary variables for some fixed maximum clause length p and maximum appearance q of a variable, [38]. Furthermore, Positive-Not-All-Equal-2-Flip, a reformulation of MaxCut, is known to be $\mathcal{PLS}$-complete along with Maximum 2-Satisfiability, [57]. For any generalized Satisfiability local search problem, a dichotomy theorem states that the problem is either in $\mathcal{P}$ or $\mathcal{PLS}$-complete, [13].

4.2 Recent Results and the Connection to Game Theory

Recently, the field of local search has attracted additional attention from game theory considering the complexity of computing a pure Nash equilibrium. In this chapter, we only consider pure Nash equilibria; thus, we omit pure for sake of readability. In a Nash equilibrium, no player can unilaterally deviate and strictly decrease his private cost. This gives rise to a simple dynamics to compute Nash equilibria, known as *Nash dynamics*: In each round, a single player is allowed to perform a *selfish step*, i.e., unilaterally change his strategy and strictly improve his private cost; the dynamics terminates once no such player exists. In the special dynamics known as *best-response dynamics*, each player selects a strategy which maximizes the decrease of his private cost. Proving the existence of Nash equilibria for classes of games is usually done using some potential function argument. A potential function maps every state of the game to a positive integer such that every selfish step of each player decreases the potential function. Local optima of potential functions then coincide with *Nash equilibria*. In case of polynomial time computable potential functions, the problem of finding

a Nash equilibrium can be formulated as a $\mathcal{PLS}$-problem, where the neighborhood structure is superimposed by the Nash dynamics. In this subsection, we survey results considering the complexity of computing a Nash equilibrium in CongestionGames. For $\mathcal{PLS}$-completeness results in other games, we refer the reader to [3,20,28].

The Class of Congestion Games. Inspired by road traffic and more recently the internet, the class of CongestionGames has been under severe scrutiny for the last years. CongestionGames are games where n weighted users myopically select strategies s_i from their set of strategies S_i (subsets of the set of shared resources $\mathcal{R}$) such that their individual delay on all resources in the current strategy profile $\mathsf{s} = (s_1, \ldots, s_n)$ is minimized. Here, the individual delay of a player is the sum of the delays on each resource the player is using. The delay on resource $r \in \mathcal{R}$ is the value of the delay function $d_r : \mathbb{N} \mapsto \mathbb{N}$ for the sum of the weights of the players using the resource. A congestion game is unweighted, if the weights of all players are equal. In this case, $n_r(\mathsf{s})$ denotes the number of players using resource $r \in \mathcal{R}$ in strategy profile s. In network congestion games, the set of strategies for each player corresponds to all his source-sink paths. A congestion game is symmetric if all players have the same set of strategies and asymmetric otherwise. Unweighted congestion games are known to possess an exact potential function $\Phi(\mathsf{s}) = \sum_{r \in \mathcal{R}} \sum_{i=1}^{n_r(\mathsf{s})} d_r(i)$, [56], hence, computing a Nash equilibrium can be formulated as finding a minimum of the potential function. Note that if all delay functions are polynomials, then the Nash dynamics converges after at most a polynomial number of iterations.

Unweighted Congestion Games. First, we consider unweighted congestion games. In their seminal paper, Fabrikant et al., [22], settle the complexity of computing a Nash equilibrium. Finding a Nash equilibrium in symmetric network congestion games is polynomial time solvable by reduction to the min-cost-flow problem, [22]. Computing a Nash equilibrium in symmetric congestion games and in asymmetric network congestion games is tight $\mathcal{PLS}$-complete; hence, these games possess the all-exp property, [22]. Notably, neither the numbers of players nor the number of resources is bounded in both reductions. Despite the polynomial time computability of Nash equilibria in symmetric network congestion games, an asymmetric network congestion game having the all-exp property can be embedded; hence, the class of symmetric network congestion games possess the all-exp property, [3]. The $\mathcal{PLS}$-completeness result for asymmetric network congestion games was refined to hold for undirected networks even if all latency functions are linear, including an elegant proof for the $\mathcal{PLS}$-completeness of directed asymmetric network congestion games, [3]. If the strategy space of each player consists of the bases of a matroid over the set of resources, then Nash equilibria can be efficiently computed using a the best-response Nash dynamics. The matroid property is a sufficient and necessary condition on the combinatorial structure of the players' strategy spaces to guarantee fast convergence to Nash equilibria, [3]. On the other hand, if the strategies of a congestion game

fulfill a so called (1,2)-exchange property, then the problem of finding a Nash Equilibrium has the is-exp property, [3].

Approximate Nash Equilibria in Congestion Games. Since in many cases, the computation of Nash equilibria is as hard as finding a local optimum for any problem in $\mathcal{PLS}$, one might hope that the computation of approximate Nash equilibria yields a significant performance improvement. A δ-approximate Nash equilibrium is a strategy profile in which no player can unilaterally improve his delay by a factor of at least δ. The existence of a δ-approximate Nash equilibrium is guaranteed by Rosenthal's potential function and may be computed by adapting the Nash dynamics to now incorporate unilateral deviations which improve the delay of the deviating player by a factor of at least δ. The hope for an efficient computation of δ-approximate Nash equilibria might be additionally fueled by the existence of an FPTAS for finding approximate local optima for all linear combinatorial optimization problem in $\mathcal{PLS}$, [48]. In particular, the authors provided a rule of choosing improving steps that lead to an approximate solution within a polynomial number of steps. Unfortunately this notion of approximation is not sufficient for approximate Nash equilibria, since each selfish player is ignorant of the potential function. There might be solutions which are up to a factor of δ close to a local optimum considering the potential function, but a single player might still have an incentive of δ or more to unilaterally deviate.

As outlined above, if all delay functions are polynomials, then Nash equilibria can be efficiently computed. The question arises to which extent the delay functions can be generalized such that polynomial time convergence to approximate Nash equilibria is still guaranteed. A congestion game satisfies the α-bounded jump condition, if for every resource with at least one player, the addition of a single player increases the delay by at most a factor of α. Note that delay functions satisfying the α-bounded jump condition are more general than polynomial delay functions, but not exponential. This condition does not trim the inherent complexity of congestion games, as finding a Nash equilibrium in symmetric congestion games satisfying the α-bounded jump condition with $\alpha = 2$ is $\mathcal{PLS}$-complete, [15]. On the other hand, computing a δ-approximate Nash equilibrium in symmetric congestion games, where each edge satisfies the α-bounded jump condition, is polynomial time computable, since the sequence of slightly restricted δ-selfish steps converges within a number of steps that is polynomial in the number of players, α, and δ^{-1}. The problem of computing a Nash equilibria in an asymmetric congestion games satisfying the α-bounded jump condition via δ-best improvement steps possess the all-exp property, [60]. Hence, the positive result is restricted to symmetric congestion games. Computing a δ-approximate Nash equilibrium in arbitrary congestion games is $\mathcal{PLS}$-complete, for every polynomial time computable approximation factor δ, [60]. As this reduction is tight, the problem of computing a δ-approximate Nash equilibrium in arbitrary congestion games has the all-exp property.

Player-Specific (Singleton) Congestion Games. Now, we turn our attention to a variation of congestion games known as congestion games with *player-specific*

latency functions, originally introduced by Milchtaich, [45]. While in congestion games, all players share the same delay function for a resource, player-specific delay functions rather emphasize the players' personal preferences for certain resources. Note that this setting also allows to model that each player may only use a certain subset of the resources. In the model of *singleton games*, every strategy of each player consists of a single resource. These games are a subclass of network congestion games, where players route their demand through a simple network of parallel edges between two nodes s and t.

First, we consider singleton congestion games with player-specific latency functions. In general, unweighted singleton congestion games with player-specific delay functions possess Nash equilibria, but may not necessarily have a potential function, even in the case of three players, [45]. Moreover, this result is tight since in the case of two players, the Nash dynamics converges, [45]. In the case of weighted players and player-specific non-decreasing latency functions, Nash equilibria might not exist for games with three players, [45]. Unweighted singleton congestion games with player-specific linear delay functions without a constant term possess a potential function and hence Nash equilibria, [26]. This result does not extend to player-specific linear delay functions. Concatenations of two unweighted singleton congestion possess Nash equilibria, [46], but may not necessarily have a potential function, [26]. The special class of unweighted singleton congestion games with player-specific constants, where all player-specific delay functions are composed of a common resource-specific delay function, but each player may have a player-specific constant for the particular resource, possesses a potential function, [44]. Computing a Nash equilibrium in a symmetric network congestion game with player-specific constants is $\mathcal{PLS}$-hard, [44]. Restricted congestion games are congestion games with player-specific constants zero or infinity. For restricted network congestion games with three players, finding a Nash equilibrium is $\mathcal{PLS}$-complete, [4]. On the other hand, computing Nash equilibria in restricted singleton congestion games can be efficiently done if all delay function are the identity function, [25]. Notably, all delay functions are required to be the identity function; for arbitrary linear delay functions, the complexity is unknown.

Singleton Congestion Games. Here, we only consider games where for each resource, all players share the same latency function. In case all latency functions are non-decreasing, Nash equilibria are guaranteed to exist, [56]. If all latency functions are linear, then Nash equilibria can be efficiently computed using Graham's LPT algorithm, [24,29]. Notably, these games also possess the is-exp property, even for the best-response Nash dynamics, [23]. The simplicity of the network seems in most cases to allow for efficient algorithms to compute Nash equilibria, but the situation somewhat changes, once weighted players may form arbitrary non-fixed coalitions. If players may form such coalitions of size at most 8 and in each improving step of a coalition, the maximum cost of its members decreases, then computing a Nash equilibrium is $\mathcal{PLS}$-complete, [19].

Related: Equilibrium search and $\mathcal{PPAD}$. In the previous sections we have seen that the search for Nash Equilibria in CongestionGames can be treated as a

local search problem because Rosenthal's potential function provides a measure by which Selfish Steps become improving steps with respect to the potential. However, there are famous problems which ask for a Nash Equilibrium of a strategic game for which no such potential function is known. Among them is BIMATRIX, i.e. the problem of computing a Nash Equilibrium of a 2-player game with rational utilities for the players.

An algorithm that computes a Nash Equilibrium for 2-player games was already developed in 1964 by Lemke and Howson, [40], which works similarly to the simplex algorithm and is comparably successful in practical application. However, it was shown that there are 2-player games, for which the Lemke-Howson algorithm even in the best case takes an exponential number of steps before reaching a Nash Equilibrium, [54]. Motivated by proofs that show the existence of solutions via the lemma that "every graph has an even number of odd degree nodes" Papadimitriou invented the complexity class $\mathcal{PPAD}$, [51]. The problems in this class are defined via implicitly given, exponentially large directed graphs consisting of directed paths, cycles, and single nodes, where one artificial source is known. The problems asks for an endpoint of a path, i.e. a source or a sink, that is different from the artificial source. Already in [58] it was shown that the possible steps of the Lemke-Howson algorithm induce a graph with the above properties and therefore BIMATRIX $\in \mathcal{PPAD}$, [68].

Significant progress in the classification of the complexity of computing a Nash Equilibrium for games with a finite number of players was achieved by Daskalakis, Goldberg, and Papadimitriou, [17], who showed that computing a Nash Equilibrium for four-player games is $\mathcal{PPAD}$-hard. Using their construction Chen, Deng, and Teng, [14], even showed that BIMATRIX is $\mathcal{PPAD}$-complete. However, as for congestion games, the approximation of Nash Equilibria of 2-player games appears no easier than the computation of a Nash Equilibrium itself since the existence of an FPTAS would imply that $\mathcal{PPAD}$ is in $\mathcal{P}$, [14]. For further readings on results related to $\mathcal{PPAD}$, we refer the reader to the survey by Yannakakis, [72].

5 Open Problems

In this section, we present an excerpt of additional open problems, besides the ones we presented in the course of this paper. We categorize them as problems from combinatorial optimization, game theory, and smoothed complexity.

Combinatorial Optimization. As outlined in Section 4.1, several hardness results have been established for $\mathcal{PLS}$-problems arising from classical combinatorial optimization. While these results show that the problems are $\mathcal{PLS}$-complete for general instances, the *exact* bounds on the $\mathcal{PLS}$-complexity of the problems are still unknown. For MAXCUT, we outlined in Section 4.1 that the problem is $\mathcal{PLS}$-complete for unbounded degree and polynomial time solvable for maximum degree three. While the result of Krentel, [38], implies that MAXCUT is $\mathcal{PLS}$-complete for *some* fixed maximum degree, the minimum degree required for the problem to be $\mathcal{PLS}$-compete is still unknown. Similarly, for the TRAVELING

Salesman Problem where the neighborhood structure is superimposed by the well-known k-OPT algorithm, [41], the only published results imply the $\mathcal{PLS}$-completeness for $k \gg 1,000$. The exact complexity for $2 \leq k \ll 1,000$ is still unsettled. For the fundamental Maximum Constraint Assignment-problem, there remains a gap between known $\mathcal{PLS}$-completeness results for $(4,3,3)$-MCA and $(2,2,\cdot)$-MCA, which can be solved in polynomial time. Considering the simplex algorithm, the question arises, if there exists a pivot rule which guarantees a polynomially upper bounded number of steps of the simplex algorithm.

Game Theory. Section 4.1 outlines that except for special cases, general CongestionGames are $\mathcal{PLS}$-complete. For a thorough understanding, determining the source of the inherent complexity would be beneficial. To this extent, the impact of the players, the resources, and their mutual interaction on the hardness are yet to be determined. For example, the exact complexity of CongestionGames with a finite number of players is still unknown. As outlined in Section 4.2, results only exist for the case of restricted network congestion games. In the simplest model of restricted singleton congestion games and arbitrary non-decreasing latency functions, no results are known, considering the complexity of computing a Nash equilibrium. Turning to approximation, the complexity of computing δ-approximate Nash equilibria for symmetric CongestionGames, which do not satisfy some smoothness condition is still unsettled. Turning to coalitions in the symmetric singleton congestion game model, there remains a gap on the complexity of computing a Nash equilibrium. For coalitions of size one, the problem is known to be computable in polynomial time, while if users may form arbitrary non-fixed coalitions of size at least eight, the problem becomes $\mathcal{PLS}$-complete.

Smoothed Complexity. $\mathcal{PLS}$-completeness implies that computing locally optimal solutions for $\mathcal{PLS}$-complete problem is as hard as computing a locally optimal solution for *any* problem in the class $\mathcal{PLS}$. While loosely speaking, this states that there exist instances on which local search algorithms take exponential time, what is the smoothed complexity of $\mathcal{PLS}$-complete problems? Let us remark that smoothed complexity results only exist for problems from Section 2; for all other problems outlined in this survey, especially MaxCut and CongestionGames we are not aware of any results considering their smoothed complexity.

What's There to Take Home? Local search is a standard approach to approximate solutions of hard combinatorial optimization which has proven to be successful in a wide range of areas over the last decades. The framework of $\mathcal{PLS}$ was introduced to theoretically investigate the complexity of local search problems and drew additional attention from game theory in recent years. In general, our knowledge about the class $\mathcal{PLS}$ is currently rather limited and by far not comparable with the rich knowledge which we have about the class $\mathcal{NP}$. As we have outlined in the course of this paper, the complexity of a handful of $\mathcal{PLS}$-problems has been settled; for numerous other problems, determining their complexity remains tantalizingly open.

References

1. Aarts, E., Korst, J., Michiels, W.: Theoretical Aspects of Local Search. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag New York, Inc., Secaucus (2007)
2. Aarts, E., Lenstra, J.K. (eds.): Local Search in Combinatorial Optimization. John Wiley & Sons, Inc., New York (1997)
3. Ackermann, H., Röglin, H., Vöcking, B.: On the impact of combinatorial structure on congestion games. J. ACM 55(6), 1–22 (2008)
4. Ackermann, H., Skopalik, A.: On the complexity of pure nash equilibria in player-specific network congestion games. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 419–430. Springer, Heidelberg (2007)
5. Adler, I., Karp, R.M., Shamir, R.: A simplex variant solving an times linear program in ($\min(m^2, d^2)$expected number of pivot steps. J. Complexity 3(4), 372–387 (1987)
6. Alekseeva, E., Kochetov, Y., Plyasunov, A.: Complexity of local search for the p-median problem. European Journal of Operational Research 191(3), 736–752 (2008)
7. Arthur, D., Manthey, B., Röglin, H.: k-means has polynomial smoothed complexity. CoRR, abs/0904.1113 (2009)
8. Babai, L. (ed.): Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16. ACM, New York (2004)
9. Bixby, R.E.: Solving real-world linear programs: A decade and more of progress. Operations Research 50(1), 3–15 (2002)
10. Borgwardt, K.H.: Probabilistic analysis of simplex algorithms. Contemporary Mathematics 114, 21–34 (1990)
11. Brandt, F., Fischer, F., Holzer, M.: Symmetries and the complexity of pure nash equilibrium. Journal of Computer and System Sciences 75(3), 163–177 (2009)
12. Chandra, B., Karloff, H.J., Tovey, C.A.: New results on the old k-opt algorithm for the traveling salesman problem. SIAM J. Comput. 28(6), 1998–2029 (1999)
13. Chapdelaine, P., Creignou, N.: The complexity of boolean constraint satisfaction local search problems. Annals of Mathematics and Artificial Intelligence 43(1-4), 51–63 (2005)
14. Chen, X., Deng, X., Teng, S.-H.: Settling the complexity of computing two-player nash equilibria. J. ACM 56(3) (2009)
15. Chien, S., Sinclair, A.: Convergence to approximate nash equilibria in congestion games. Games and Economic Behavior (2009), (in press, accepted manuscript)
16. Danzig, G.: Programming in linear structure. Technical report, U.S. Air Force Comptroller, USAF, Washington, D.C. (1948)
17. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a nash equilibrium. SIAM J. Comput. 39(1), 195–259 (2009)
18. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley & Sons, Inc., Chichester (2000)
19. Dumrauf, D., Monien, B.: On the road to $\mathcal{PLS}$-completeness: 8 agents in a singleton congestion game. In: Papadimitriou, C.H., Zhang, S. (eds.) WINE. LNCS, vol. 5385, pp. 94–108. Springer, Heidelberg (2008)
20. Dunkel, J., Schulz, A.S.: On the complexity of pure-strategy nash equilibria in congestion and local-effect games. Math. Oper. Res. 33(4), 851–868 (2008)
21. Englert, M., Röglin, H., Vöcking, B.: Worst case and probabilistic analysis of the 2-opt algorithm for the tsp. Electronic Colloquium on Computational Complexity (ECCC) 13(092) (2006)

22. Fabrikant, A., Papadimitriou, C.H., Talwar, K.: The complexity of pure nash equilibria. In: Babai [8], pp. 604–612
23. Feldmann, R., Gairing, M., Lücking, T., Monien, B., Rode, M.: Nashification and the coordination ratio for a selfish routing game. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 514–526. Springer, Heidelberg (2003)
24. Fotakis, D., Kontogiannis, S.C., Koutsoupias, E., Mavronicolas, M., Spirakis, P.G.: The structure and complexity of nash equilibria for a selfish routing game. In: Widmayer, P., Triguero Ruiz, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 123–134. Springer, Heidelberg (2002)
25. Gairing, M., Lücking, T., Mavronicolas, M., Monien, B.: Computing nash equilibria for scheduling on restricted parallel links. In: Babai [8], pp. 613–622
26. Gairing, M., Monien, B., Tiemann, K.: Routing (un-) splittable flow in games with player-specific linear latency functions. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4051, pp. 501–512. Springer, Heidelberg (2006)
27. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. Mathematical Sciences Series. W. H. Freeman & Co., New York (1990)
28. Goemans, M.X., Mirrokni, V.S., Vetta, A.: Sink equilibria and convergence. In: FOCS, pp. 142–154. IEEE Computer Society, Los Alamitos (2005)
29. Graham, R.L.: Bounds on multiprocessing timing anomalies. SIAM Journal on Applied Mathematics 17(2), 416–429 (1969)
30. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Comput. Surv. 31(3), 264–323 (1999)
31. Johnson, D.S., McGeoch, L.A.: The Traveling Salesman Problem: A Case Study. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 215–310. Wiley and Sons, New York (1997)
32. Johnson, D.S., Papadimtriou, C.H., Yannakakis, M.: How easy is local search? Journal of Computer and System Science 37(1), 79–100 (1988)
33. Kalai, G., Kleitman, D.J.: A quasi-polynomial bound for the diameter of graphs of polyhedra. American Mathematical Socity 26(2), 315–316 (1992)
34. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica 4(4), 373–396 (1984)
35. Khachiyan, L.G.: A polynomial algorithm in linear programming. In: Doklady Akademia Nauk SSSR, pp. 1093–1096 (1979)
36. Klauck, H.: On the hardness of global and local approximation. In: Karlsson, R., Lingas, A. (eds.) SWAT 1996. LNCS, vol. 1097, pp. 88–99. Springer, Heidelberg (1996)
37. Kochetov, Y., Ivanenko, D.: Computationally difficult instances for the uncapacitated facility location problem. In: Proceedings MIC 2003 (2003)
38. Krentel, M.W.: Structure in locally optimal solutions (extended abstract). In: FOCS, pp. 216–221. IEEE, Los Alamitos (1989)
39. Krentel, M.W.: On finding and verifying locally optimal solutions. SIAM Journal on Computing 19(4), 742–749 (1990)
40. Lemke Jr., C.E., Howsen, J.T.: Equilibrium points of bimatrix games. SIAM Journal on Applied Mathematics 12(2), 413–423 (1964)
41. Lin, S.: Computer solutions of the traveling salesman problem. Bell System Technical Journal 44, 2245–2269 (1965)

42. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Operations Research 21(2), 498–516 (1973)
43. Lueker, G.S.: Manuscript. Princeton University, Princeton (1975)
44. Mavronicolas, M., Milchtaich, I., Monien, B., Tiemann, K.: Congestion games with player-specific constants. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 633–644. Springer, Heidelberg (2007)
45. Milchtaich, I.: Congestion games with player-specific payoff functions. Games and Economic Behavior 13(1), 111–124 (1996)
46. Milchtaich, I.: The equilibrium existence problem in finite network congestion games. In: Spirakis, P.G., Mavronicolas, M., Kontogiannis, S.C. (eds.) WINE 2006. LNCS, vol. 4286, pp. 87–98. Springer, Heidelberg (2006)
47. Monien, B., Tscheuschner, T.: On the power of nodes of degree four in the local max-cut problem. In: Diaz, J. (ed.) Proceedings of the 7th International Conference on Algorithms and Complexity, Rome, Italy (2010)
48. Orlin, J.B., Punnen, A.P., Schulz, A.S.: Approximate local search in combinatorial optimization. In: Ian Munro, J. (ed.) Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms (SODA), pp. 587–596. SIAM, Philadelphia (2004)
49. Papadimitriou, C.H., Schäffer, A.A., Yannakakis, M.: On the complexity of local search. In: STOC 1990: Proceedings of the twenty-second annual ACM symposium on Theory of computing, pp. 438–445. ACM Press, New York (1990)
50. Papadimitriou, C.H.: The complexity of the lin-kernighan heuristic for the traveling salesman problem. SIAM J. Comput. 21(3), 450–465 (1992)
51. Papadimitriou, C.H.: On the complexity of the parity argument and other inefficient proofs of existence. J. Comput. Syst. Sci. 48(3), 498–532 (1994)
52. Poljak, S.: Integer linear programs and local search for max-cut. SIAM J. Comput. 24(4), 822–839 (1995)
53. Prokopyev, O.A., Huang, H.-X., Pardalos, P.M.: On complexity of unconstrained hyperbolic 0-1 programming problems. Operations Research Letters 33(3), 312–318 (2005)
54. von Stengel, B., Savani, R.: Hard-to-solve bimatrix games. Econometrica 74, 397–429 (2006)
55. Reinelt, G.: Tsplib - a traveling salesman problem library. INFORMS Journal on Computing 3(4), 376–384 (1991)
56. Rosenthal, R.W.: A class of games possessing pure-strategy nash equilibria. International Journal of Game Theory 2, 65–67 (1973)
57. Schäffer, A.A., Yannakakis, M.: Simple local search problems that are hard to solve. SIAM J. Comput. 20(1), 56–87 (1991)
58. Shapley, L.S.: A note on the lemke-howson algorithm. Pivoting and Extension 1, 175–189 (2009)
59. Shimozono, S.: Finding optimal subgraphs by local search. Theoretical Computer Science 172(1-2), 265–271 (1997)
60. Skopalik, A., Vöcking, B.: Inapproximability of pure nash equilibria. In: Ladner, R.E., Dwork, C. (eds.) STOC, pp. 355–364. ACM, New York (2008)
61. Smale, S.: On the average number of steps in the simplex method of linear programming. Mathematical Programming 27, 241–262 (1983)
62. Roberto, S.-O.: Local search. In: Gonzalez, T.F. (ed.) Handbook of Approximation Algorithms and Metaheuristics. Chapman & Hall/Crc Computer & Information Science Series, Chapman & Hall/CRC (2007)
63. Spielman, D.A., Teng, S.-H.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. J. ACM 51(3), 385–463 (2004)

64. Todd, M.J.: The many facets of linear programming. Mathematical Programming 91(3), 417–436 (2001)
65. Kleinschmidt, P., Klee, V.: The d-step conjecture and its relatives. Mathematics of Operations Research 12(4), 718–755 (1987)
66. Vattani, A.: k-means requires exponentially many iterations even in the plane. In: Hershberger, J., Fogel, E. (eds.) Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, pp. 324–332 (2009)
67. Minty, G.J., Klee, V.: How good is the simplex algorithm? In: Inequalities III, pp. 159–175. Academic Press, New York (1972)
68. von Stengel, B.: Computing equilibria for two-person games. In: Handbook of game theory, vol. 3, pp. 1726–1759 (2002)
69. Vredeveld, T., Lenstra, J.K.: On local search for the generalized graph coloring problem. Operations Research Letters 31(1), 28–34 (2003)
70. Yannakakis, M.: Computational complexity. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 19–55. Wiley, Chichester (1997)
71. Yannakakis, M.: The analysis of local search problems and their heuristics. In: Choffrut, C., Lengauer, T. (eds.) STACS 1990. LNCS, vol. 415, pp. 298–311. Springer, Heidelberg (1990)
72. Yannakakis, M.: Equilibria, fixed points, and complexity classes. Computer Science Review 3(2), 71–85 (2009)

When Conflicting Constraints Can Be Resolved – The Lovász Local Lemma and Satisfiability

Emo Welzl

ETH Zurich
Switzerland
emo@inf.ethz.ch

Abstract. The general theme underlying the talk is the following question: Given a set of constraints, how much interleaving of these constraints (relative to how serious these constraints are) is necessary so that all of them cannot be satisfied simultaneously?

For a concrete setting we consider boolean formulas in conjunctive normal form (CNF), where the clauses play the role of constraints. If all clauses are large, it needs many clauses to obtain an unsatisfiable formula; moreover, these clauses have to interleave. We review quantitative results for the amount of interleaving required, many of which rely on the Lovász Local Lemma, a probabilistic lemma with many applications in combinatorics.

In positive terms, we are interested in simple combinatorial conditions which guarantee for a CNF formula to be satisfiable. The criteria obtained are nontrivial in the sense that even though they are easy to check, it is by far not obvious how to compute a satisfying assignment efficiently in case the conditions are fulfilled; until recently, it was not known how to do so. It is also remarkable that while deciding satisfiability is trivial for formulas that satisfy the conditions, a slightest relaxation of the conditions leads us into the territory of NP-completeness.

This is a survey with recent results by Heidi Gebauer, Robin Moser, Dominik Scheder, and Gábor Tardos, among others.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, p. 18, 2010.

Plane Spanners of Maximum Degree Six

Nicolas Bonichon[1,⋆], Cyril Gavoille[1,⋆,⋆⋆],
Nicolas Hanusse[1,⋆], and Ljubomir Perković[2,⋆⋆⋆]

[1] Laboratoire Bordelais de Recherche en Informatique (LaBRI),
Université de Bordeaux, France
{bonichon,gavoille,hanusse}@labri.fr
[2] School of Computing, DePaul University, USA
lperkovic@cs.depaul.edu

Abstract. We consider the question: "What is the smallest degree that can be achieved for a plane spanner of a Euclidean graph $\mathcal{E}$?" The best known bound on the degree is 14. We show that $\mathcal{E}$ always contains a plane spanner of maximum degree 6 and stretch factor 6. This spanner can be constructed efficiently in linear time given the Triangular Distance Delaunay triangulation introduced by Chew.

1 Introduction

In this paper we focus on the following question:

> "What is the smallest maximum degree that can be achieved for plane spanners of the complete, two-dimensional Euclidean graph $\mathcal{E}$?"

This question happens to be Open Problem 14 in a very recent survey of plane geometric spanners [BS]. It is an interesting, fundamental question that has curiously not been studied much. (Unbounded degree) plane spanners have been studied extensively: obtaining a tight bound on the stretch factor of the Delaunay graph is one of the big open problems in the field. Dobkin *et al.* [ADDJ90] were the first to prove that Delaunay graphs are (plane) spanners. The stretch factor they obtained was subsequently improved by Keil & Gutwin [KG92] as shown in Table 1. The plane spanner with the best known upper bound on the stretch factor is not the Delaunay graph however, but the TD-Delaunay graph introduced by Chew [Che89] whose stretch factor is 2 (see Table 1). We note that the Delaunay and the TD-Delaunay graphs may have unbounded degree.

Just as (unbounded degree) plane spanners, bounded degree (but not necessarily planar) spanners of $\mathcal{E}$ have been well studied and are, to some extent, well understood: it is known that spanners of maximum degree 2 do not exist in general and that spanners of maximum degree 3 can always be constructed (Das & Heffernan [DH96]). In recent years, bounded degree plane spanners have

⋆ Supported by the ANR-project "ALADDIN", the ANR-project "GRAAL", and the équipe-projet INRIA "CÉPAGE".
⋆⋆ Member of the "Institut Universitaire de France".
⋆⋆⋆ Supported by a DePaul University research grant.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 19–30, 2010.

Table 1. Results on plane spanners with maximum degree bounded by Δ

paper	Δ	stretch factor
Dobkin *et al.* [ADDJ90]	∞	$\frac{\pi(1+\sqrt{5})}{2} \approx 5.08$
Keil & Gutwin [KG92]	∞	$C_0 = \frac{4\pi\sqrt{3}}{9} \approx 2.42$
Chew [Che89]	∞	2
Bose *et al.* [BGS05]	27	$(\pi+1)C_0 \approx 10.016$
Li & Wang [LW04]	23	$(1+\pi\sin\frac{\pi}{4})C_0 \approx 7.79$
Bose *et al.* [BSX09]	17	$(2+2\sqrt{3}+\frac{3\pi}{2}+2\pi\sin(\frac{\pi}{12}))C_0 \approx 28.54$
Kanj & Perković [KP08]	14	$(1+\frac{2\pi}{14\cos(\frac{\pi}{14})})C_0 \approx 3.53$
This paper: Section 3	9	6
This paper: Section 4	6	6

been used as the building block of wireless network communication topologies. Emerging wireless distributed system technologies such as wireless ad-hoc and sensor networks are often modeled as proximity graphs in the Euclidean plane. Spanners of proximity graphs represent topologies that can be used for efficient unicasting, multicasting, *and/or* broadcasting. For these applications, spanners are typically required to be planar and have bounded degree: the planarity requirement is for efficient routing, while the bounded degree requirement is due to the physical limitations of wireless devices.

Bose *et al.* [BGS05] were the first to show how to extract a spanning subgraph of the Delaunay graph that is a bounded-degree, plane spanner of $\mathcal{E}$. The maximum degree and stretch factor they obtained was subsequently improved by Li & Wang [LW04], Bose *et al.* [BSX09], and by Kanj & Perković [KP08] (see all bounds in Table 1). The approach used in all of these results was to extract a bounded degree spanning subgraph of the *classical Delaunay triangulation*. The main goal in this line of research was to obtain a bounded-degree plane spanner of $\mathcal{E}$ with the *smallest possible stretch factor*.

In this paper we propose a new goal and a new approach. Our goal is to obtain a plane spanner with the *smallest possible maximum degree*. We believe this question is fundamental. The best known bound on the degree of a plane spanner is 14 [KP08]. In some wireless network applications, such a bound is too high. Bluetooth scatternets, for example, can be modeled as spanners of $\mathcal{E}$ where master nodes must have at most 7 slave nodes [LSW04].

Our approach consists of two steps. We first extract a maximum degree 9 spanning subgraph H_2 from Chew's *TD-Delaunay graph* instead of the classical Delaunay graph. Graph H_2 is a spanner of the TD-Delaunay graph of stretch factor 3, and thus a spanner of $\mathcal{E}$ of stretch factor 6. With this fact, combined with a recent result of [BGHI10], we derive *en passant* the following: Every Θ_6-graph contains a spanner of maximum degree 6 that has stretch factor 3. Secondly, by the use of local modifications of H_2, we show how to decrease the maximum degree from 9 to 6 without increasing the maximum stretch while preserving planarity.

Our approach leads to a significant improvement in the maximum degree of the plane spanner, from 14 down to 6 (see Table 1). Just as the Delaunay graph,

the TD-Delaunay graph of a set of n points in the plane can be computed in time $O(n \log n)$ [Che89]. Given this graph, our final spanner H_4 can be constructed in $O(n)$ time. We note that our analysis of the stretch factor of the spanner is tight: we can place points in the plane so that the resulting degree 6 spanner has stretch factor arbitrarily close to 6.

2 Preliminaries

Given points in the two-dimensional Euclidean plane, the complete Euclidean graph $\mathcal{E}$ is the complete weighted graph embedded in the plane whose nodes are identified with the points. In the following, given a graph G, $V(G)$ and $E(G)$ stand for the set of nodes and edges of G. For every pair of nodes u and w, we identify with edge uw the segment $[uw]$ and associate an edge length equal to the Euclidean distance $|uw|$. We say that a subgraph H of a graph G is a *t-spanner* of G if for any pair of vertices u, v of G, the distance between u and v in H is at most t times the distance between u and v in G; the constant t is referred to as the *stretch factor* of H (with respect to G). We will say that H is a spanner if it is a t-spanner of $\mathcal{E}$ for some constant t.

A *cone* C is the region in the plane between two rays that emanate from the same point. Let us consider the rays obtained by a rotation of the positive x-axis by angles of $i\pi/3$ with $i = 0, 1, \ldots, 5$. Each pair of successive rays defines a cone whose apex is the origin. Let $\mathcal{C}_6 = (\overline{C}_2, C_1, \overline{C}_3, C_2, \overline{C}_1, C_3)$ be the sequence of cones obtained, in counter-clockwise order, starting from the positive x-axis. The cones C_1, C_2, C_3 are said to be *positive* and the cones $\overline{C}_1, \overline{C}_2, \overline{C}_3$ are said to be *negative*. We assume a cyclic structure on the labels so that $i+1$ and $i-1$ are always defined. For a positive cone C_i, the clockwise next cone is the negative cone $\overline{C}_{i+1}$ and the counter-clockwise next cone is the negative cone $\overline{C}_{i-1}$.

For each cone $C \in \mathcal{C}_6$, let ℓ_C be the bisector ray of C (in Figure 1, for example, the bisector rays of the positive cones are shown). For each cone C and each point u, we define $C^u := \{x + u : x \in C\}$, the translation of cone C from the origin to point u. We set $\mathcal{C}_6^u := \{C + u : C \in \mathcal{C}_6\}$, the set of all six cones at u. Observe that $w \in C_i^u$ if and only if $u \in \overline{C}_i^w$.

Let v be a point in a cone C^u. The *projection distance* from u to v, denoted $d_P(u, v)$, is the Euclidean distance between u and the projection of v onto ℓ_{C^u}.

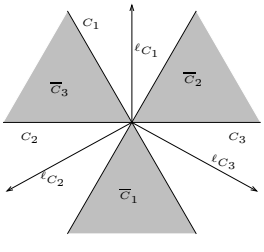


Fig. 1. Illustration of notations used for describing cones. Positive cones are white and negative cones are grey. Bisector rays of the three positive cones are shown.

For any two points v and w in C^u, v is *closer* to u than w if and only if $d_P(u,v) < d_P(u,w)$. We denote by $parent_i(u)$ the closest point from u belonging to cone C_i^u.

We say that a given set of points S are in *general position* if no two points of S form a line parallel to one of the rays that define the cones of $\mathcal{C}_6$. For the sake of simplicity, in the rest of the paper we only consider sets of points that are in general position. This will imply that it is impossible that two points v and w have equal projective distance from another point u. Note that, in any case, ties can be broken arbitrarily when ordering points that have the same distance (for instance, using a counter-clockwise ordering around u).

Our starting point is a geometric graph proposed in [BGHI10]. It represents the first step of our construction.

Step 1. *Every node u of $\mathcal{E}$ chooses $parent_i(u)$ in each non-empty cone C_i^u. We denote by H_1 the resulting subgraph.*

While we consider H_1 to be undirected, we will refer to an edge in H_1 as *outgoing* with respect to u when chosen by u and *incoming* with respect to $v = parent_i(u)$, and we color it i if it belongs to C_i^u. Note that edge uv is in the negative cone $\overline{C}_i^v$ of v.

Theorem 1 ([BGHI10]). *The subgraph H_1 of $\mathcal{E}$:*

- *is a plane graph such that every face (except the outerface) is a triangle,*
- *is a 2-spanner of $\mathcal{E}$, and*
- *has at most one (outgoing) edge in every positive cone of every node.*

Note that the number of incoming edges at a particular node of H_1 is not bounded.

In our construction of the subsequent subgraph H_2 of H_1, for every node u some neighbors of u will play an important role. Given i, let $children_i(u)$ be the set of points v such that $u = parent_i(v)$. Note that $children_i(u) \subseteq \overline{C}_i^u$. In $children_i(u)$, three special points are named:

- $closest_i(u)$ is the closest point of $children_i(u)$;
- $first_i(u)$ is the first point of $children_i(u)$ in counter-clockwise order starting from x axis;
- $last_i(u)$ is the last point of $children_i(u)$ in counter-clockwise order starting from x axis.

Note that some of these nodes can be undefined if the cone $\overline{C}_i^u$ is empty. Let (u,v) be an edge such that $v = parent_i(u)$. A node w is *i-relevant* with respect to (wrt) u if $w \in \overline{C}_i^v = \overline{C}_i^{parent_i(u)}$, and either $w = first_{i-1}(u) \neq closest_{i-1}(u)$, or $w = last_{i+1}(u) \neq closest_{i+1}(u)$. When node w is defined as $first_{i-1}(u)$ or $last_{i+1}(u)$, we will omit specifying "with respect to u". For instance, in Figure 2 **(a)**, the vertices v_l and v_r are i-relevant with respect to w. In Figure 2 **(b)** the vertex $v_r = last_{i+1}(w)$ is not i-relevant since it is not in $\overline{C}_i^u$ and $v_l = first_{i-1}(w)$ is not i-relevant since it is also $closest_{i-1}(w)$.

3 A Simple Planar 6-Spanner of Maximum Degree 9

In this section we describe the construction of H_2, a plane 6-spanner of $\mathcal{E}$ of maximum degree 9. The construction of H_2 is very simple and can be easily distributed:

Step 2. *Let H_2 be the graph obtained by choosing edges of H_1 as follows: for each node u and each negative cone $\overline{C}_i^u$:*

- *add edge $(u, closest_i(u))$ if $closest_i(u)$ exists,*
- *add edge $(u, first_i(u))$ if $first_i(u)$ exists and is $(i+1)$-relevant and*
- *add edge $(u, last_i(u))$ if $last_i(u)$ exists and is $(i-1)$-relevant.*

Note that H_2 is a subgraph of H_1 that is easily seen to have maximum degree no greater than 12 (there are at most 3 incident edges per negative cone and 1 incident edge per positive cone). Surprisingly, we shall prove that:

Theorem 2. *The graph H_2*

- *has maximum degree 9,*
- *is a 3-spanner of H_1, and thus a 6-spanner of $\mathcal{E}$.*

The remainder of this section is devoted to proving this theorem.

The charge of a cone. In order to bound the degree of a node in H_2, we devise a counting scheme. Each edge incident to a node is *charged to* some cone of that node as follows:

- each negative cone $\overline{C}_i^u$ is charged by the edge $(u, closest_i(u))$ if $closest_i(u)$ exists.
- each positive cone C_i^u is charged by $(u, parent_i(u))$ if this edge is in H_2, by edge $(u, first_{i-1}(u))$ if $first_{i-1}(u)$ is i-relevant, and by $(u, last_{i+1}(u))$ if $last_{i+1}(u)$ is i-relevant.

For instance, in Figure 2 **(a)**, the cone C_i^w is charged to twice: once by $v_l w$ and once by $v_r w$; the cone $\overline{C}_{i-1}^w$ is charged to once by its smallest edge. In **(b)**, the cone C_i^w is not charged to at all: $v_l w$ is the shortest edge in $\overline{C}_{i-1}^w$. In **(c)** the cone C_i^w is charged to once by $v_l w$ and once by the edge wu.

We will denote by $charge(C)$ the charge to cone C. With the counting scheme in place, we can prove the following lemma, which implies the first part of Theorem 2, since the sum of charges to cones of a vertex is equal to its degree in H_2.

Lemma 1. *Each negative cone of every node has at most 1 edge charged to it and each positive cone of every node has at most 2 edges charged to it.*

Proof. Since a negative cone never has more than one edge charged to it, all we need to do is to argue that no positive cone has 3 edges charged to it. Let C_i^w be a positive cone at some node w.

Let $u = parent_i(w)$. If the edge (w, u) is not in H_2 then clearly $charge(C_i^w) \leq 2$. Otherwise, we consider three cases:

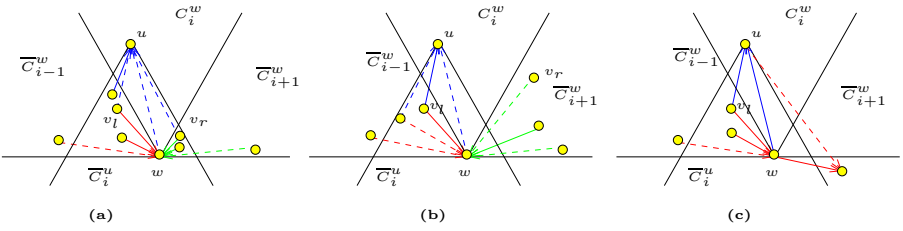


Fig. 2. In all three cases, edge wu is in H_1 but $w \neq closest_i(u)$. Solid edges are edges that are in H_2. **(a)** The edges wv_l and wv_r are, respectively, the clockwise last in C_{i-1}^w and clockwise first in C_{i+1}^w and are i-relevant with respect to w. **(b)** The edge wv_r is not i-relevant because wv_r is not in $\overline{C}_i^u$. The edge wv_l is in H_2 but is not i-relevant because it is the shortest edge in $\overline{C}_{i-1}^w$. **(c)** Edge wv_l is i-relevant. Note that edge wu is in H_2 because it is $(i-1)$-relevant with respect to u.

Case 1: $w = closest_i(u)$. Any point of $R = \overline{C}_i^u \cap \{\overline{C}_{i-1}^w \cup \overline{C}_{i+1}^w\}$ is closer to u than w. Since w is the closest neighbor of u in $\overline{C}_i^u$ the region R is empty. Hence the nodes $first_{i-1}(w)$ and $last_{i+1}(w)$ are not i-relevant. Hence $charge(C_i^w) = 1$.

Case 2: $w = last_i(u)$ and w is $(i-1)$-relevant (with respect to u, see Figure 2 **(c)**). In this case, w, u and $parent_{i-1}(w) = parent_{i-1}(u)$ form an empty triangle in H_1. Therefore, $\overline{C}_i^u \cap \overline{C}_{i+1}^w$ is empty. Hence $last_{i+1}(w)$ is not i-relevant. Hence $charge(C_i^w) \leq 2$.

Case 3: $w = first_i(u)$ and w is $(i+1)$-relevant. Using an argument symmetric to the one in Case 2, $\overline{C}_i^u \cap \overline{C}_{i-1}^w$ is empty. Hence $first_{i-1}(w)$ is not i-relevant. Hence $charge(C_i^w) \leq 2$. □

The above proof gives additional structural information that we will use in the next section:

Corollary 1. *Let $u = parent_i(w)$. If $charge(C_i^w) = 2$ then either:*

1. *(w, u) is not in H_2, and $first_{i-1}(w)$ and $last_{i+1}(w)$ are i-relevant (and are therefore neighbors of w in H_2), or*
2. *$w = last_i(u)$ is $(i-1)$-relevant and $first_{i-1}(w)$ is i-relevant (and thus (w, u) and $(first_{i-1}(w), w)$ are in H_2), or*
3. *$w = first_i(u)$ is $(i+1)$-relevant and $last_{i+1}(w)$ is i-relevant (and thus (w, u) and $(last_{i+1}(w), w)$ are in H_2).*

In case 1 above, note that nodes $first_{i-1}(w)$, w, and $last_{i+1}(w)$ are both in $\overline{C}_i^u$ and that u is closer from both $first_{i-1}(w)$ and $last_{i+1}(w)$ than from w. When the case 1 condition holds, we say that w is *i-distant*.

In order to prove that H_2 is a 3-spanner of H_1, we need to show that for every edge wu in H_1 but not in H_2 there is a path from u to w in H_2 whose length is at most $3|uw|$. Let wu be an incoming edge of H_1 with respect to u. Since $wu \notin H_2$, the shortest incoming edge of H_1 in the cone C of u containing wu must be in H_2: we call it vu. Without loss of generality, we assume vu is clockwise from wu with respect to u.

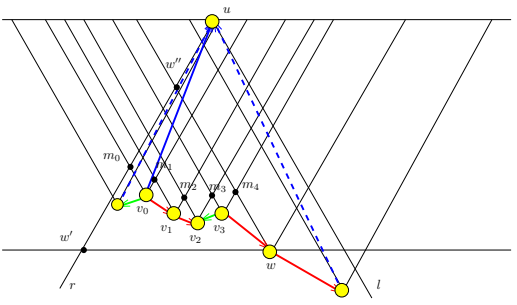


Fig. 3. Canonical path

We consider all the edges of H_1 incident to u that are contained in the cone $\overline{C}_i^u$ and lying in-between vu and wu, and we denote them, in counter-clockwise order, $vu = v_0u$, v_1u, ..., $v_ku = wu$. Because H_1 is a triangulation, the path v_0v_1, v_1v_2, ..., $v_{k-1}v_k$ is in H_1. We call this path the *canonical path with respect to u and w* (see Figure 3). Note that the order – u first, w second – matters.

Lemma 2. *Let $(w, u = parent_i(w))$ be an edge of H_1 and $v = closest_i(u)$. If $w \neq v$ then:*

1. *H_2 contains the edge vu and the canonical path with respect to u and w*
2. $|uv| + \sum_{i=1}^{k} |v_{i-1}v_i| \leq 3|uw|$.

The second part of Theorem 2 follows because Lemma 2 shows that for every wu in H_1 but not in H_2, if wu is incoming with respect to u then the path consisting of uv and the canonical path with respect to u and w will exist in H_2 and the length of this path is at most $3|uw|$.

Proof (of Lemma 2). Let $e = (v_j, v_{j+1})$ be an edge of the canonical path with respect to u and w. First assume that e is incoming at v_j. Observe that v_{j+1} is the neighbor of v_j that is just before u in the counter-clockwise ordering of neighbors around v_j in the triangulation H_1. Hence $v_{j+1} = last_{i+1}(v_j)$. Since v_{j+1} is in $\overline{C}_i^u$, v_{j+1} is i-relevant (with respect to v_j) or $v_{j+1} = closest_{i+1}(v_j)$. In both cases, e is in H_2. Now assume that the edge e is incoming at v_{j+1}. We similarly prove that $v_j = first_{i-1}(v_{j+1})$ and that v_j is i-relevant (with respect to v_{j+1}) or $v_j = closest_{i-1}(v_{j+1})$. In both cases, e is in H_2. This proves the first part of the lemma.

In order to prove the second part of Lemma 2, we denote by $C_i^{v_i}$ the cone containing u of canonical path node v_i, for $i = 0, 1, ..., k$. We denote by r_i and l_i the rays defining the clockwise and counter-clockwise boundaries of cone $C_i^{v_i}$. Let r and l be the rays defining the clockwise and counter-clockwise boundaries of cone $\overline{C}_i^u$. We define the point m_o as the intersection of half-lines r and l_0, points m_i as the intersections of half-lines r_{i-1} and l_i for every $1 \leq i \leq k$. Let w' be the intersection of the half-line r and the line orthogonal to ℓ_C $(C = C_i^w)$

passing through w, and let w'' be the intersection of half-lines l_k and r (see Figure 3).

We note that $|uv| = |uv_0| \leq |um_0| + |m_0v_0|$, and

$$|v_{i-1}v_i| \leq |v_{i-1}m_i| + |m_iv_i|$$

for every $1 \leq i \leq k$. Also $|uv_0| \geq |um_0|$. Then

$$\begin{aligned} |uv_0| + \sum_{i=1}^{k} |v_{i-1}v_i| &\leq |um_0| + \sum_{i=0}^{k} |m_iv_i| + \sum_{i=0}^{k-1} |v_im_{i+1}| \\ &\leq |um_0| + |ww''| + |w''m_0| \\ &\leq |uw'| + |ww'| + |w''w'| \\ &\leq |uw'| + 2|ww'| \end{aligned}$$

Observe that $|uw| = \sqrt{(|uw'|\cos\pi/6)^2 + (|ww'| - |uw'|/2)^2}$. Let $\alpha = |ww'|/|uw'|$; note that $0 \leq \alpha \leq 1$. Then

$$\begin{aligned} \frac{|uw'| + 2|ww'|}{|uw|} &\leq \frac{(1+2\alpha)|uw'|}{\sqrt{(|uw'|\cos\pi/6)^2 + ((\alpha - 1/2)|uw'|)^2}} \\ &\leq \frac{1+2\alpha}{\sqrt{1-\alpha+\alpha^2}} \\ &\leq \max_{\alpha\in[0..1]} \left\{ \frac{1+2\alpha}{\sqrt{1-\alpha+\alpha^2}} \right\} \\ &\leq 3 \end{aligned}$$

□

4 A Planar 6-Spanner of Maximum Degree 6

We now carefully delete edges from and add other edges to H_2, in order to decrease the maximum degree of the graph to 6 while maintaining the stretch factor. We do that by attempting to decrease the number of edges charged to a positive cone down to 1. We will not be able to do so for some cones. We will show that we can amortize the positive charge of 2 for such cones over a neighboring negative cone with charge 0. By Corollary 1, we only need to take care of two cases (the third case is symmetric to the second).

Before presenting our final construction, we start with a structural property of some positive cones in H_3 with a charge of 2. Recall that a node is *i-distant* if it has two i-relevant neighbors in H_2 (this corresponds to case 1 of Corollary 1). For instance, in Figure 3, the node v_2 is i-distant.

Lemma 3 (Forbidden charge sequence). *If, in H_2, charge$(C_i^w) = 2$ and w is not a i-distant node:*

– *either* $first_{i-1}(w)$ *is* i*-relevant,* $charge(C^w_{i-1}) \leq 1$ *and* $charge(\overline{C}^w_{i+1}) = 0$ *or*
– $last_{i+1}(w)$ *is* i*-relevant,* $charge(C^w_{i+1}) \leq 1$ *and* $charge(\overline{C}^w_{i-1}) = 0$.

Proof. By Corollary 1, if w is not an i-distant node, either $first_{i-1}(w)$ or $last_{i+1}(w)$ is i-relevant. We assume the second case, and the first will follow by symmetry.

We first prove the existence of a cone of charge 0. If $u = parent_i(w)$, then by Corollary 1, $w = last_i(u)$ and w is $(i-1)$-relevant (with respect to u). This means that nodes w, u, and $v = parent_{i-1}(u) = parent_{i-1}(w)$ form an empty triangle in H_1 and therefore there is no edge that ends up in $\overline{C}^w_{i+1}$. Hence $charge(\overline{C}^w_{i+1}) = 0$.

Let us prove now by contradiction that $charge(C^w_{i-1}) \leq 1$. Assume that $charge(C^w_{i-1}) = 2$. By Corollary 1 there can be three cases. We have just shown that there are no edges in $\overline{C}^w_{i+1}$, so there cannot be a node $first_{i+1}(w)$ and the first two cases cannot apply. Case 3 of Corollary 1 implies that $w = first_{i-1}(v)$ which is not possible because edge (u, v) is before (w, u) in the counter-clockwise ordering of edges in $\overline{C}_{i-1}(v)$. □

Step 3. *We construct* H_3 *from* H_2 *as follows: for every integer* $1 \leq i \leq 3$ *and for every* i*-distant node* w*:*

– *add the edge* $(first_{i-1}(w), last_{i+1}(w))$ *to* H_3*;*
– *let* w' *be the node among* $\{first_{i-1}(w), last_{i+1}(w)\}$ *which is greater in the canonical path order. Remove the edge* (w, w') *from* H_3.

New charge assignments. Since a new edge e is added between nodes $first_{i-1}(w)$ and $last_{i+1}(w)$ in Step 3, we assign the charge of e to $\overline{C}^{first_{i-1}(w)}_{i+1}$ and to $\overline{C}^{last_{i+1}(w)}_{i-1}$. For the sake of convenience, we denote by $\widetilde{charge}(C)$ the total charge, after Step 3, of cone C in H_3 and the next graph we will construct, H_4. The following lemma shows that the application of Step 3 does not create a cone of charge 2 and decreases the charge of cone C^w_i of i-distant node w from 2 to 1.

Lemma 4 (Distant nodes). *If* w *is an* i*-distant node then:*

– $\widetilde{charge}(C^w_i) = charge(C^w_i) - 1 = 1$*;*
– $\widetilde{charge}(\overline{C}^{first_{i-1}(w)}_{i+1}) = charge(\overline{C}^{last_{i+1}(w)}_{i-1}) = 1$.

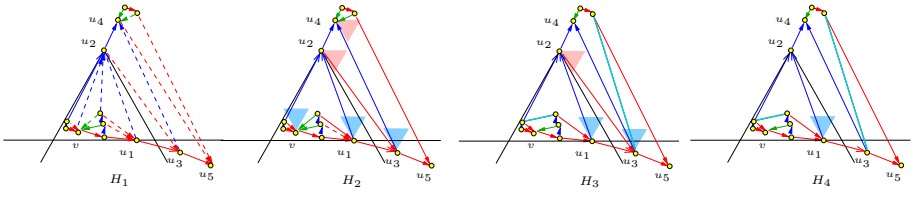


Fig. 4. From H_1 (plain arrows are the closest edges) to H_4. Light blue and pink positive cones have a charge equal to 2. The node v is i-distant and the node u_4 is $i+1$-distant.

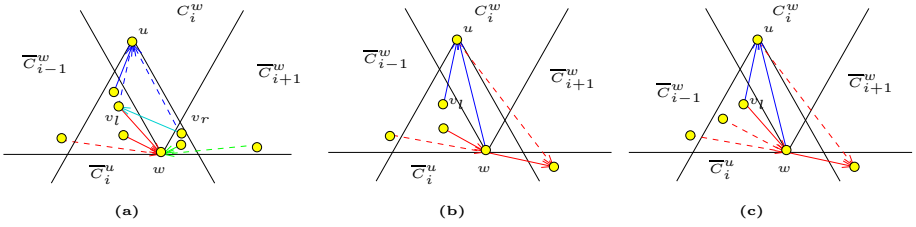


Fig. 5. (a) Step 3 applied on the configuration of Figure 2 **(a)**: the edge wv_r is removed because the canonical path of w with respect to u doesn't use it. The edge is then replaced by the edge $v_r v_l$. **(b)** Step 4 applied on the configuration of Figure 2 **(c)**: the edge wv_l is removed. **(c)** the edge $v_l w$ is the shortest in $\overline{C_{i-1}^w}$; the cone C_i^w is thus charged to only once, by wu, and the edge $v_l w$ is not removed during step 4.

Step 4. *We construct H_4 from H_3 as follows: for every integer $1 \leq i \leq 3$ and for every node w such that $\widetilde{charge}(C_i^w) = 2$ and $\widetilde{charge}(\overline{C}_{i-1}^w) = \widetilde{charge}(\overline{C}_{i+1}^w) = 1$, if $w = last_i(parent_i(w))$ then remove the edge $(w, first_{i-1}(w))$ from H_3 and otherwise remove $(w, last_{i+1}(w))$.*

Lemma 5. *There is a 1-1 mapping between each positive cone C_i^w that has charge 2 after step 4 and a negative cone at w that has charge 0.*

Proof. Corollary 1 gives the properties of two types of cones with charge 2 in H_2. If cone C_i^w is one in which w is an i-distant node in H_2, then C_i^w will have a charge of 1 after Step 3, by Lemma 4. If w is not i-distant, there can be two cases according to Lemma 3. We assume the first (the second follows by symmetry); so we assume that $\overline{C}_{i+1}^w$ has charge 0 in H_2. If that charge is increased to 1 in step 3, then step 4 will decrease the charge of C_i^w down to 1. So, if C_i^w still has a charge of 2 after step 4, then $\overline{C}_{i+1}^w$ will still has charge 0 and we map C_i^w to this adjacent negative cone. The only positive cone that could possibly map to $\overline{C}_{i+1}^w$ would be the other positive cone adjacent to $\overline{C}_{i+1}^w$, C_{i-1}^w, but that cone has charge at most 1 by Lemma 3. □

Theorem 3. *H_4 is a plane 6-spanner of $\mathcal{E}$ of maximum degree 6.*

Proof. By Corollary 1, Lemma 4, and Lemma 5, it is clear that H_4 has maximum degree 6.

Let us show H_4 is a 6-spanner of $\mathcal{E}$. By Lemma 2, for every edge wu in H_1 but not in H_2, the canonical path with respect to u and w in H_2 has total length at most $3|wu|$. We argue that the removal, in step 3, of edges on the canonical path from u is compensated by the addition of other edges in step 3. Observe first that while some edges of the canonical path may have been removed from H_2 in step 3, in every case a shortcut has been added. Some edges have also been removed in step 4. The removed edge is always the last edge on the canonical path from u to w, where uw is the first or last edge, in counterclockwise order, in some negative cone at u and $uw \in H_2$. This means that the canonical path

edge is only needed to reach w from u, and no other nodes. Therefore it can be removed since $wu \in H_2$. In summary, no "intermediate" canonical path edge is dropped without a shortcut, and "final" canonical path edges will be dropped only when no longer needed. Therefore any canonical path (of length at most $3|wu|$) in H_2 is replaced by a new path (with shortcuts) of length at most $3|wu|$. By Lemma 2, the above argument can also be directly applied for every edge $xy \in H_2$ removed in H_3.

It remains to show that H_4 is planar. More precisely we have to show that edges introduced during step 3 do not create crossings in H_3. Let $v_l v_r$ be an edge created during step 3. Observe that in H_1 there are two adjacent triangular faces $f_1 = uv_l w$ and $f_2 = uwv_r$. Since the edge wv_l is in C^w_{i-1} and v_r is in C^w_{i+1} the angle $v_l u v_r$ is less than π. Hence the edge $v_l v_r$ is inside the two faces f_1 and f_2. The only edge of H_1 that crosses the edge $v_l v_r$ is the edge wu. Since the edge wu is not present in H_4 there is no crossing between an edge of $H_1 \cap H_3$ and an edge added during step 3. What now remains to be done is to show that two edges added during step 3 cannot cross each other. Let $v'_l v'_r$ be an edge created during step 3 and let $f'_1 = u'v'_l w'$ and $f'_2 = u'w'v'_r$ the two faces of H_1 containing this edge. If the edges $v_l v_r$ and $v'_l v'_r$ cross each other, then they are supported by at least one common face of H_1, i.e. $\{f_1, f_2\} \cap \{f'_1, f'_2\} \neq \emptyset$. Observe that the edges $v_l u, wu$ and $v_r u$ are colored i, the edge wv_l is colored $i-1$ and the edge wv_r is colored $i+1$. Similarly the edges $v'_l u', w'u'$ and $v'_r u'$ are colored i', the edge $w'v'_l$ is colored $i'-1$ and the edge $w'v'_r$ is colored $i'+1$. Each face f_1, f_2, f'_1 and f'_2 has two edges of the same color, hence $i = i'$. Because of the color of the third edge of each face, this implies that $f_1 = f'_1$ and $f_2 = f'_2$, and so $v_l v_r = v'_l v'_r$. This shows that H_4 has no crossing. □

5 Conclusion

Our construction can be used to obtain a spanner of the unit-hexagonal graph, a generalization of the complete Euclidean graph. More precisely, every unit-hexagonal graph G has a spanner of maximum degree 6 and stretch factor 6. This can be done by observing that, in our construction, the canonical path associated with each edge $e \in G \setminus H_2$ is composed of edges of "length" at most the "length" of e, where the "length" of e is the hexagonal-distance[1] between its end-points.

References

[ADDJ90] Althöfer, I., Das, G., Dobkin, D.P., Joseph, D.: Generating sparse spanners for weighted graphs. In: SWAT, pp. 26–37 (1990)

[BGHI10] Bonichon, N., Gavoille, C., Hanusse, N., Ilcinkas, D.: Connections between Theta-graphs, Delaunay triangulations, and orthogonal surfaces. Technical Report hal-00454565, HAL (February 2010)

[1] The hexagonal-distance between u and v Euclidean distance between u and the projection of v onto the bisector of the cone of $\mathcal{C}^u_6$ where v belongs to.

[BGS05] Bose, P., Gudmundsson, J., Smid, M.: Constructing plane spanners of bounded degree and low weight. Algorithmica 42(3-4), 249–264 (2005)

[BS] Bose, P., Smid, M.: On plane geometric spanners: A survey and open problems (submitted)

[BSX09] Bose, P., Smid, M., Xu, D.: Delaunay and diamond triangulations contain spanners of bounded degree. International Journal of Computational Geometry and Applications 19(2), 119–140 (2009)

[Che89] Paul Chew, L.: There are planar graphs almost as good as the complete graph. Journal of Computer and System Sciences 39(2), 205–219 (1989)

[DH96] Das, G., Heffernan, P.J.: Constructing degree-3 spanners with other sparseness properties. Int. J. Found. Comput. Sci. 7(2), 121–136 (1996)

[KG92] Mark Keil, J., Gutwin, C.A.: Classes of graphs which approximate the complete Euclidean graph. Discrete & Computational Geometry 7(1), 13–28 (1992)

[KP08] Kanj, I.A., Perković, L.: On geometric spanners of Euclidean and unit disk graphs. In: 25^{th} Annual Symposium on Theoretical Aspects of Computer Science (STACS), vol. hal-00231084, pp. 409–420. HAL (February 2008)

[LSW04] Li, X.-Y., Stojmenovic, I., Wang, Y.: Partial Delaunay triangulation and degree limited localized bluetooth scatternet formation. IEEE Trans. Parallel Distrib. Syst. 15(4), 350–361 (2004)

[LW04] Li, X.-Y., Wang, Y.: Efficient construction of low weight bounded degree planar spanner. International Journal of Computational Geometry and Applications 14(1–2), 69–84 (2004)

The Positive Semidefinite Grothendieck Problem with Rank Constraint

Jop Briët[1,⋆], Fernando Mário de Oliveira Filho[2,⋆⋆] and Frank Vallentin[3,⋆⋆⋆]

[1] Centrum Wiskunde & Informatica (CWI), Science Park 123,
1098 SJ Amsterdam, The Netherlands
j.briet@cwi.nl
[2] Department of Econometrics ad OR, Tilburg University,
5000 LE Tilburg, The Netherlands
f.m.de.oliveira.filho@cwi.nl
[3] Delft Institute of Applied Mathematics, Technical University of Delft,
P.O. Box 5031, 2600 GA Delft, The Netherlands
f.vallentin@tudelft.nl

Abstract. Given a positive integer n and a positive semidefinite matrix $A = (A_{ij}) \in \mathbb{R}^{m \times m}$ the positive semidefinite Grothendieck problem with rank-n-constraint (SDP_n) is

$$\text{maximize } \sum_{i=1}^{m} \sum_{j=1}^{m} A_{ij}\, x_i \cdot x_j, \text{ where } x_1, \ldots, x_m \in S^{n-1}.$$

In this paper we design a randomized polynomial-time approximation algorithm for SDP_n achieving an approximation ratio of

$$\gamma(n) = \frac{2}{n} \left(\frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \right)^2 = 1 - \Theta(1/n).$$

We show that under the assumption of the unique games conjecture the achieved approximation ratio is optimal: There is no polynomial-time algorithm which approximates SDP_n with a ratio greater than $\gamma(n)$. We improve the approximation ratio of the best known polynomial-time algorithm for SDP_1 from $2/\pi$ to $2/(\pi\gamma(m)) = 2/\pi + \Theta(1/m)$, and we show a tighter approximation ratio for SDP_n when A is the Laplacian matrix of a weighted graph with nonnegative edge weights.

⋆ The first author is supported by Vici grant 639.023.302 from the Netherlands Organization for Scientific Research (NWO), by the European Commission under the Integrated Project Qubit Applications (QAP) funded by the IST directorate as Contract Number 015848, and by the Dutch BSIK/BRICKS project.

⋆⋆ The second author was partially supported by CAPES/Brazil under grant BEX 2421/04-6, and the research was carried out at the Centrum Wiskunde & Informatica, The Netherlands.

⋆⋆⋆ The third author is supported by Vidi grant 639.032.917 from the Netherlands Organization for Scientific Research (NWO).

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 31–42, 2010.

1 Introduction

Given a positive integer n and a positive semidefinite matrix $A = (A_{ij}) \in \mathbb{R}^{m\times m}$, the *positive semidefinite Grothendieck problem with rank-n-constraint* is defined as

$$\mathrm{SDP}_n(A) = \max\left\{\sum_{i=1}^{m}\sum_{j=1}^{m} A_{ij}\, x_i \cdot x_j : x_1, \ldots, x_m \in S^{n-1}\right\},$$

where $S^{n-1} = \{x \in \mathbb{R}^n : x \cdot x = 1\}$ is the unit sphere; the inner product matrix of the vectors $x_1, \ldots, x_m$ has rank n. This problem was introduced by Briët, Buhrman, and Toner [5] in the context of quantum nonlocality where they applied it to nonlocal XOR games. The case $n = 1$ is the classical positive semidefinite Grothendieck problem where $x_1, \ldots, x_m \in \{-1, +1\}$. It was introduced by Grothendieck [7] in the study of norms of tensor products of Banach spaces. It is an NP-hard problem: If A is the Laplacian matrix of a graph then $\mathrm{SDP}_1(A)$ coincides with the value of a maximum cut of the graph. The maximum cut problem (MAX CUT) is one of Karp's 21 NP-complete problems. Over the last years, there has been a lot of work on algorithmic applications, interpretations and generalizations of the Grothendieck problem and the companion Grothendieck inequalities. For instance, Nesterov [18] showed that it has applications to finding and analyzing semidefinite relaxations of nonconvex quadratic optimization problems. Ben-Tal and Nemirovski [4] showed that it has applications to quadratic Lyapunov stability synthesis in system and control theory. Alon and Naor [3] showed that it has applications to constructing Szemerédi partitions of graphs and to estimating the cut norms of matrices. Linial and Shraibman [15] showed that it has applications to find lower bounds in communication complexity. Khot and Naor [12], [13] showed that it has applications to kernel clustering. See also Alon, Makarychev, Makarychev, and Naor [2], and Raghavendra and Steurer [20].

One can reformulate the positive semidefinite Grothendieck problem with rank-n-constraint as a semidefinite program with an additional rank constraint:

$$\begin{aligned}
\text{maximize } & \sum_{i=1}^{m}\sum_{j=1}^{m} A_{ij}X_{ij} \\
\text{subject to } & X = (X_{ij}) \in \mathbb{R}^{m\times m} \text{ is positive semidefinite,} \\
& X_{ii} = 1, \text{ for } i = 1, \ldots, m, \\
& X \text{ has rank at most } n.
\end{aligned}$$

When n is a constant that does not depend on the matrix size m there is no polynomial-time algorithm known which solves SDP_n. It is also not known if the problem SDP_n is NP-hard when $n \geq 2$. On the other hand the *semidefinite relaxation* of $\mathrm{SDP}_n(A)$ defined by

$$\mathrm{SDP}_\infty(A) = \max\left\{\sum_{i=1}^{m}\sum_{j=1}^{m} A_{ij}\, u_i \cdot u_j : u_1, \ldots, u_m \in S^{\infty}\right\}$$

can be computed in polynomial time using semidefinite programming. Here S^∞ denotes the unit sphere of the Hilbert space $l^2(\mathbb{R})$ of square summable sequences, which contains $\mathbb{R}^n$ as the subspace of the first n components. Clearly, it would suffice to use unit vectors in $\mathbb{R}^m$ for solving $\mathrm{SDP}_\infty(A)$ when $A \in \mathbb{R}^{m\times m}$, but using S^∞ will simplify many formulations in this paper. Rietz [21] (in the context of the Grothendieck inequality) and Nesterov [18] (in the context of approximation algorithms for NP-hard problems) showed that SDP_1 and SDP_∞ are always within a factor of at most $2/\pi$ from each other. That is, for all positive semidefinite matrices $A \in \mathbb{R}^{m\times m}$ we have

$$1 \geq \frac{\mathrm{SDP}_1(A)}{\mathrm{SDP}_\infty(A)} \geq \frac{2}{\pi}. \tag{1}$$

By exhibiting an explicit series of positive semidefinite matrices, Grothendieck [7] (see also Alon and Naor [3, Section 5.2]) showed that one cannot improve the constant $2/\pi$ to $2/\pi+\varepsilon$ for any positive ε which is independent of m. Nesterov [18] gave a randomized polynomial-time approximation algorithm for SDP_1 with approximation ratio $2/\pi$ which can be derandomized using the techniques presented by Mahajan and Ramesh [16]. This algorithm is optimal in the following sense: Khot and Naor [12] showed that under the assumption of the unique games conjecture (UGC) there is no polynomial-time algorithm which approximates SDP_1 by a constant $2/\pi + \varepsilon$ for any positive ε independent of m. The unique games conjecture was introduced by Khot [10] and by now many tight UGC hardness results are known, see e.g. Khot, Kindler, Mossel, and O'Donnell [11] for the maximum cut problem, Khot and Regev [14] for the minimum vertex cover problem, and Raghavendra [19] for general constrained satisfaction problems. The aim of this paper is to provide a corresponding analysis for SDP_n.

Our Results

In Section 2 we start by reviewing our methodological contributions: Our main contribution is the analysis of a rounding scheme which can deal with rank-n-constraints in semidefinite programs. For this we use the Wishart distribution from multivariate statistics (see e.g. Muirhead [17]). We believe this analysis is of independent interest and will turn out to be useful in different contexts, e.g. for approximating low dimensional geometric embeddings. Our second contribution is that we improve the constant in inequality (1) slightly by considering functions of positive type for the unit sphere S^{m-1} and applying a characterization of Schoenberg [22]. This slight improvement is the key for our UGC hardness result of approximating SDP_n given in Theorem 3. We analyze our rounding scheme in Section 3.

Theorem 1. *For all positive semidefinite matrices $A \in \mathbb{R}^{m\times m}$ we have*

$$1 \geq \frac{\mathrm{SDP}_n(A)}{\mathrm{SDP}_\infty(A)} \geq \gamma(n) = \frac{2}{n}\left(\frac{\Gamma((n+1)/2)}{\Gamma(n/2)}\right)^2 = 1 - \Theta(1/n),$$

and there is a randomized polynomial-time approximation algorithm for SDP_n achieving this ratio.

The first three values of $\gamma(n)$ are:

$$\gamma(1) = 2/\pi = 0.63661\ldots$$
$$\gamma(2) = \pi/4 = 0.78539\ldots$$
$$\gamma(3) = 8/(3\pi) = 0.84882\ldots$$

In Section 4 we show that one can improve inequality (1) slightly:

Theorem 2. *For all positive semidefinite matrices $A \in \mathbb{R}^{m\times m}$ we have*

$$1 \geq \frac{\mathrm{SDP}_1(A)}{\mathrm{SDP}_\infty(A)} \geq \frac{2}{\pi\gamma(m)} = \frac{m}{\pi}\left(\frac{\Gamma(m/2)}{\Gamma((m+1)/2)}\right)^2 = \frac{2}{\pi} + \Theta\left(\frac{1}{m}\right),$$

and there is a polynomial-time approximation algorithm for SDP_1 *achieving this ratio.*

With this, the current complexity status of the problem SDP_1 is similar to the one of the minimum vertex cover problem. Karakostas [9] showed that one can approximate the minimum vertex cover problem for a graph having vertex set V with an approximation ratio of $2-\Theta(1/\sqrt{\log|V|})$ in polynomial time. On the other hand, Khot and Regev [14] showed, assuming the unique games conjecture, that there is no polynomial-time algorithm which approximates the minimum vertex cover problem with an approximation factor of $2-\varepsilon$ for any positive ε which is independent of $|V|$. In Section 5 we show that the approximation ratio $\gamma(n)$ given in Theorem 1 is optimal for SDP_n under the assumption of the unique games conjecture. By using the arguments of the proof of Theorem 2 and by the UGC hardness of approximating SDP_1 due to Khot and Naor [12] we get the following tight UGC hardness result for approximating SDP_n.

Theorem 3. *Under the assumption of the unique games conjecture there is no polynomial-time algorithm which approximates* SDP_n *with an approximation ratio greater than $\gamma(n)+\varepsilon$ for any positive ε which is independent of the matrix size m.*

In Section 6 we show that a better approximation ratio can be achieved when the matrix A is the Laplacian matrix of a graph with nonnegative edge weights.

2 Rounding Schemes and Functions of Positive Type

In this section we discuss our rounding scheme which rounds an optimal solution of SDP_∞ to a feasible solution of SDP_n. In the case $n=1$ our rounding scheme is equivalent to the classical scheme of Goemans and Williamson [6]. To analyze the rounding scheme we use functions of positive type for unit spheres. The randomized polynomial-time approximation algorithm which we use in the proofs of the theorems is the following three-step process. The last two steps are our rounding scheme.

1. Solve $\mathrm{SDP}_\infty(A)$, obtaining vectors $u_1, \ldots, u_m \in S^{m-1}$.
2. Choose $X = (X_{ij}) \in \mathbb{R}^{n \times m}$ so that every matrix entry X_{ij} is distributed independently according to the standard normal distribution with mean 0 and variance 1: $X_{ij} \sim N(0,1)$.
3. Set $x_i = Xu_i / \|Xu_i\| \in S^{n-1}$ with $i = 1, \ldots, m$.

The quality of the feasible solution $x_1, \ldots, x_m$ for SDP_n is measured by the expectation

$$\mathbb{E}\left[\sum_{i=1}^m \sum_{j=1}^m A_{ij}\, x_i \cdot x_j\right] = \sum_{i=1}^m \sum_{j=1}^m A_{ij} \mathbb{E}\left[\frac{Xu_i}{\|Xu_i\|} \cdot \frac{Xu_j}{\|Xu_j\|}\right],$$

which we analyze in more detail.

For vectors $u, v \in S^\infty$ we define

$$E_n(u,v) = \mathbb{E}\left[\frac{Xu}{\|Xu\|} \cdot \frac{Xv}{\|Xv\|}\right], \tag{2}$$

where $X = (X_{ij})$ is a matrix with n rows and infinitely many columns whose entries are distributed independently according to the the standard normal distribution. Of course, if $u, v \in S^{m-1}$, then it suffices to work with finite matrices $X \in \mathbb{R}^{n \times m}$.

The first important property of the expectation E_n is that it is *invariant under* $\mathrm{O}(\infty)$, i.e. for every m it is invariant under the orthogonal group $\mathrm{O}(m) = \{T \in \mathbb{R}^{m \times m} : T^\mathsf{T} T = I_m\}$, where I_m denotes the identity matrix, i.e. for every m and every pair of vectors $u, v \in S^{m-1}$ we have

$$E_n(Tu, Tv) = E_n(u,v) \quad \text{for all } T \in \mathrm{O}(m).$$

If $n = 1$, then

$$E_1(u,v) = \mathbb{E}[\operatorname{sign}(\xi \cdot u) \operatorname{sign}(\xi \cdot v)],$$

where $\xi \in \mathbb{R}^m$ is chosen at random from the m-dimensional standard normal distribution. By Grothendieck's identity (see e.g. [8, Lemma 10.2])

$$\mathbb{E}[\operatorname{sign}(\xi \cdot u) \operatorname{sign}(\xi \cdot v)] = \frac{2}{\pi} \arcsin u \cdot v.$$

Hence, the expectation E_1 only depends on the inner product $t = u \cdot v$. For general n, the $\mathrm{O}(\infty)$ invariance implies that this is true also for E_n.

The second important property of the expectation E_n (now interpreted as a function of the inner product) is that it is a function of positive type for S^∞, i.e. it is of positive type for any unit sphere S^{m-1}, independent of the dimension m. In general, a continuous function $f : [-1,1] \to \mathbb{R}$ is called *a function of positive type for* S^{m-1} if the matrix $(f(v_i \cdot v_j))_{1 \le i,j \le N}$ is positive semidefinite for every positive integer N and every choice of vectors $v_1, \ldots, v_N \in S^{m-1}$. The expectation E_n is of positive type for S^∞ because one can write it as a sum of squares. Schoenberg

[22] characterized the continuous functions $f : [-1, 1] \to \mathbb{R}$ which are of positive type for S^∞: They are of the form

$$f(t) = \sum_{i=0}^{\infty} f_i t^i,$$

with nonnegative f_i and $\sum_{i=0}^{\infty} f_i < \infty$. In the case $n = 1$ we have the series expansion

$$E_1(t) = \frac{2}{\pi} \arcsin t = \frac{2}{\pi} \sum_{i=0}^{\infty} \frac{(2i)!}{2^{2i}(i!)^2(2i+1)} t^{2i+1}.$$

In Section 3 we treat the cases $n \geq 2$.

Suppose we develop the expectation $E_n(t)$ into the series $E_n(t) = \sum_{i=0}^{\infty} f_i t^i$. Then because of Schoenberg's characterization the function $t \mapsto E_n(t) - f_1 t$ is of positive type for S^∞ as well. This together with the inequality $\sum_{i,j} X_{ij} Y_{ij} \geq 0$, which holds for all positive semidefinite matrices $X, Y \in \mathbb{R}^{m \times m}$, implies

$$\mathrm{SDP}_n(A) \geq \sum_{i=1}^{m} \sum_{j=1}^{m} A_{ij} E_n(u_i, u_j) \geq f_1 \sum_{i=1}^{m} \sum_{j=1}^{m} A_{ij}\ u_i \cdot u_j = f_1\, \mathrm{SDP}_\infty(A). \quad (3)$$

When $n = 1$ the series expansion of E_1 gives $f_1 = 2/\pi$ and the above argument is essentially the one of Nesterov [18]. To improve on this (and in this way to improve the constant $2/\pi$ in inequality (1)) one can refine the analysis by working with functions of positive type which depend on the dimension m. In Section 4 we show that $t \mapsto 2/\pi(\arcsin t - t/\gamma(m))$ is a function of positive type for S^{m-1}. For the cases $n \geq 2$ we show in Section 3 that $f_1 = \gamma(n)$

3 Analysis of the Approximation Algorithm

In this section we show that the expectation E_n defined in (2) is a function of positive type for S^∞ and that in the series expansion $E_n(t) = \sum_{i=0}^{\infty} f_i t^i$ one has $f_1 = \gamma(n)$. These two facts combined with the discussion in Section 2 imply Theorem 1. Let $u, v \in S^{m-1}$ be unit vectors and let $X = (X_{ij}) \in \mathbb{R}^{n \times m}$ be a random matrix whose entries are independently sampled from the standard normal distribution. Because of the invariance under the orthogonal group, for computing $E_n(u, v)$ we may assume that u and v are of the form

$$u = (\cos\theta, \sin\theta, 0, \ldots, 0)^\mathsf{T}$$
$$v = (\cos\theta, -\sin\theta, 0, \ldots, 0)^\mathsf{T}.$$

Then by the double-angle formula $\cos 2\theta = t$ with $t = u \cdot v$.

We have

$$Xu = \begin{pmatrix} X_{11} & X_{12} \\ \vdots & \vdots \\ X_{n1} & X_{n2} \end{pmatrix} \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \quad Xv = \begin{pmatrix} X_{11} & X_{12} \\ \vdots & \vdots \\ X_{n1} & X_{n2} \end{pmatrix} \begin{pmatrix} \cos\theta \\ -\sin\theta \end{pmatrix}.$$

Hence,

$$\frac{Xu}{\|Xu\|} \cdot \frac{Xv}{\|Xv\|} = \frac{x^\mathsf{T} Y y}{\sqrt{(x^\mathsf{T} Y x)(y^\mathsf{T} Y y)}},$$

where $x = (\cos\theta, \sin\theta)^\mathsf{T}$, $y = (\cos\theta, -\sin\theta)^\mathsf{T}$, and $Y \in \mathbb{R}^{2\times 2}$ is the Gram matrix of the two vectors $(X_{11}, \ldots, X_{n1})^\mathsf{T}$, $(X_{12}, \ldots, X_{n2})^\mathsf{T} \in \mathbb{R}^n$. By definition, Y is distributed according to the Wishart distribution from multivariate statistics. This distribution is defined as follows (see e.g. Muirhead [17]). Let p and q be positive integers so that $p \geq q$. The (standard) *Wishart distribution* $W_q(p)$ is the probability distribution of random matrices $Y = X^\mathsf{T} X \in \mathbb{R}^{q\times q}$, where the entries of the matrix $X = (X_{ij}) \in \mathbb{R}^{p\times q}$ are independently chosen from the standard normal distribution $X_{ij} \sim N(0,1)$. The density function of $Y \sim W_q(p)$ is

$$\frac{1}{2^{pq/2}\Gamma_q(p/2)} e^{-\operatorname{Tr}(Y)/2} (\det Y)^{(p-q-1)/2},$$

where Γ_q is the *multivariate gamma function*, defined as

$$\Gamma_q(x) = \pi^{q(q-1)/4} \prod_{i=1}^{q} \Gamma\Big(x - \frac{i-1}{2}\Big).$$

We denote the cone of positive semidefinite matrices of size $q \times q$ by $S^q_{\geq 0}$. In our case $p = n$ and $q = 2$. We can write $E_n(t)$ as

$$E_n(t) = \frac{1}{2^n \Gamma_2(n/2)} \int_{S^2_{\geq 0}} \frac{x^\mathsf{T} Y y}{\sqrt{(x^\mathsf{T} Y x)(y^\mathsf{T} Y y)}} e^{-\operatorname{Tr}(Y)/2} (\det Y)^{(n-3)/2} dY,$$

where $t = \cos 2\theta$, and x as well as y depend on θ. The parameterization of the cone $S^2_{\geq 0}$ given by

$$S^2_{\geq 0} = \left\{ Y = \begin{pmatrix} \frac{a}{2} + \alpha\cos\phi & \alpha\sin\phi \\ \alpha\sin\phi & \frac{a}{2} - \alpha\cos\phi \end{pmatrix} : \phi \in [0, 2\pi], \alpha \in [0, a/2], a \in \mathbb{R}_{\geq 0} \right\}$$

allows us to write the integral in a more explicit form. With this parametrization we have

$$\operatorname{Tr}(Y) = a, \quad \det(Y) = \frac{a^2}{4} - \alpha^2, \quad dY = \alpha \, d\phi d\alpha da,$$

and

$$\begin{aligned} x^\mathsf{T} Y y &= \frac{at}{2} + \alpha\cos\phi, \\ x^\mathsf{T} Y x &= \frac{a}{2} + \alpha(t\cos\phi + 2\sin\theta\cos\theta\sin\phi), \\ y^\mathsf{T} Y y &= \frac{a}{2} + \alpha(t\cos\phi - 2\sin\theta\cos\theta\sin\phi). \end{aligned}$$

So,

$$E_n(t) = \frac{1}{2^n \Gamma_2(n/2)} \int_0^\infty \int_0^{a/2} \int_0^{2\pi} \frac{\frac{at}{2} + \alpha\cos\phi}{\sqrt{(\frac{a}{2} + \alpha t\cos\phi)^2 - \alpha^2(1-t^2)(\sin\phi)^2}} \cdot e^{-a/2} \left(\frac{a^2}{4} - \alpha^2\right)^{(n-3)/2} \alpha \, d\phi d\alpha da.$$

Substituting $\alpha = (a/2)r$ and integrating over a yields

$$E_n(t) = \frac{\Gamma(n)}{2^{n-1}\Gamma_2(n/2)} \int_0^1 \int_0^{2\pi} \frac{(t + r\cos\phi) r (1-r^2)^{(n-3)/2}}{\sqrt{(1+rt\cos\phi)^2 - r^2(1-t^2)(\sin\phi)^2}} d\phi dr.$$

Using Legendre's duplication formula (see [1, Theorem 1.5.1]) $\Gamma(2x)\Gamma(1/2) = 2^{2x-1}\Gamma(x)\Gamma(x+1/2)$ one can simplify

$$\frac{\Gamma(n)}{2^{n-1}\Gamma_2(n/2)} = \frac{n-1}{2\pi}.$$

Recall from (3) that the approximation ratio is given by the coefficient f_1 in the series expansion $E_n(t) = \sum_{i=0}^{\infty} f_i t^i$. Now we compute f_1:

$$\begin{aligned} f_1 &= \frac{\partial E_n}{\partial t}(0) \\ &= \frac{n-1}{2\pi} \int_0^1 \int_0^{2\pi} \frac{r(1-r^2)^{(n-1)/2}}{(1-r^2(\sin\phi)^2)^{3/2}} d\phi dr. \end{aligned}$$

Using Euler's integral representation of the hypergeometric function [1, Theorem 2.2.1] and by substitution we get

$$\begin{aligned} f_1 &= \frac{n-1}{2\pi} \int_0^{2\pi} \frac{\Gamma(1)\Gamma((n+1)/2)}{2\Gamma((n+3)/2)} {}_2F_1\left(\begin{matrix} 3/2, 1 \\ (n+3)/2 \end{matrix}; \sin^2\phi\right) d\phi \\ &= \frac{n-1}{4\pi} \frac{\Gamma((n+1)/2)}{\Gamma((n+3)/2)} 4 \int_0^1 {}_2F_1\left(\begin{matrix} 3/2, 1 \\ (n+3)/2 \end{matrix}; t^2\right) (1-t^2)^{-1/2} dt \\ &= \frac{n-1}{\pi} \frac{\Gamma((n+1)/2)}{\Gamma((n+3)/2)} \frac{1}{2} \int_0^1 {}_2F_1\left(\begin{matrix} 3/2, 1 \\ (n+3)/2 \end{matrix}; t\right) (1-t)^{-1/2} t^{-1/2} dt. \end{aligned}$$

This simplies futher by Euler's generalized integral [1, (2.2.2)], and Gauss's summation formula [1, Theorem 2.2.2]

$$\begin{aligned} f_1 &= \frac{n-1}{2\pi} \frac{\Gamma((n+1)/2)}{\Gamma((n+3)/2)} \frac{\Gamma(1/2)\Gamma(1/2)}{\Gamma(1)} {}_3F_2\left(\begin{matrix} 3/2, 1, 1/2 \\ (n+3)/2, 1 \end{matrix}; 1\right) \\ &= \frac{n-1}{2} \frac{\Gamma((n+1)/2)}{\Gamma((n+3)/2)} {}_2F_1\left(\begin{matrix} 3/2, 1/2 \\ (n+3)/2 \end{matrix}; 1\right) \\ &= \frac{n-1}{2} \frac{\Gamma((n+1)/2)}{\Gamma((n+3)/2)} \frac{\Gamma((n+3)/2)\Gamma((n-1)/2)}{\Gamma(n/2)\Gamma((n+2)/2)} \\ &= \frac{2}{n} \left(\frac{\Gamma((n+1)/2)}{\Gamma(n/2)}\right)^2. \end{aligned}$$

4 Improved Analysis

Nesterov's proof of inequality (1) relies on the fact that the function $t \mapsto 2/\pi(\arcsin t - t)$ is of positive type for S^∞. Now we determine the largest

value $c(m)$ so that the function $t \mapsto 2/\pi(\arcsin t - c(m)t)$ is of positive type for S^{m-1} with dimension m fixed. By this we improve the approximation ratio of the algorithm given in Section 2 for SDP_1 from $2/\pi$ to $(2/\pi)c(m)$. The following lemma showing $c(m) = 1/\gamma(m)$ implies Theorem 2.

Lemma 1. *The function*

$$t \mapsto \frac{2}{\pi}\left(\arcsin t - \frac{t}{\gamma(m)}\right)$$

is of positive type for S^{m-1}.

Proof. We equip the space of all continuous functions $f : [-1, 1] \to \mathbb{R}$ with the inner product

$$(f, g)_\alpha = \int_{-1}^{1} f(t)g(t)(1-t^2)^\alpha dt,$$

where $\alpha = (m-3)/2$. With this inner product the Jacobi polynomials satisfy the orthogonality relation

$$(P_i^{(\alpha,\alpha)}, P_j^{(\alpha,\alpha)})_\alpha = 0, \quad \text{if } i \neq j,$$

where $P_i^{(\alpha,\alpha)}$ is the Jacobi polynomial of degree i with parameters (α, α), see e.g. Andrews, Askey, and Roy [1]. Schoenberg [22] showed that a continuous function $f : [-1, 1] \to \mathbb{R}$ is of positive type for S^{m-1} if and only if it is of the form

$$f(t) = \sum_{i=0}^{\infty} f_i P_i^{(\alpha,\alpha)}(t),$$

with nonnegative coefficients f_i and $\sum_{i=0}^{\infty} f_i < \infty$.

Now we interpret arc sine as a function of positive type for S^{m-1} where m is fixed. By the orthogonality relation and because of Schoenberg's result the function $\arcsin t - c(m)t$ is of positive type for S^{m-1} if and only if

$$(\arcsin t - c(m)t, P_i^{(\alpha,\alpha)})_\alpha \geq 0, \quad \text{for all } i = 0, 1, 2, \ldots$$

We have $P_1^{(\alpha,\alpha)}(t) = (\alpha+1)t$. By the orthogonality relation and because the arc sine function is of positive type we get, for $i \neq 1$,

$$(\arcsin t - c(m)t, P_i^{(\alpha,\alpha)})_\alpha = (\arcsin t, P_i^{(\alpha,\alpha)})_\alpha \geq 0.$$

This implies that the maximum $c(m)$ such that $\arcsin t - c(m)t$ is of positive type for S^{m-1} is given by $c(m) = (\arcsin t, t)_\alpha/(t, t)_\alpha$.

The numerator of $c(m)$ equals

$$\begin{aligned}(\arcsin t, t)_\alpha &= \int_{-1}^{1} \arcsin(t)t(1-t^2)^\alpha dt \\ &= \int_{-\pi/2}^{\pi/2} \theta \sin\theta(\cos\theta)^{2\alpha+1} d\theta \\ &= \frac{\Gamma(1/2)\Gamma(a+3/2)}{(2\alpha+2)\Gamma(\alpha+2)}.\end{aligned}$$

The denominator of $c(m)$ equals

$$(t,t)_\alpha = \int_{-1}^{1} t^2(1-t^2)^\alpha dt = \frac{\Gamma(3/2)\Gamma(\alpha+1)}{\Gamma(\alpha+5/2)},$$

where we used the beta integral (see e.g. Andrews, Askey, and Roy [1, (1.1.21)])

$$\int_0^1 t^{2x-1}(1-t^2)^{y-1}dr = \int_0^{\pi/2} (\sin\theta)^{2x-1}(\cos\theta)^{2y-1}d\theta = \frac{\Gamma(x)\Gamma(y)}{2\Gamma(x+y)},$$

Now, by using the functional equation $x\Gamma(x) = \Gamma(x+1)$, the desired equality $c(m) = 1/\gamma(m)$ follows. □

5 Hardness of Approximation

Proof (of Theorem 3). Suppose that ρ is the largest approximation ratio a polynomial-time algorithm can achieve for SDP_n. Let $u_1, \dots, u_m \in S^{n-1}$ be an approximate solution to $\mathrm{SDP}_n(A)$ coming from such a polynomial-time algorithm. Then,

$$\sum_{i=1}^m \sum_{j=1}^m A_{ij}\ u_i \cdot u_j \geq \rho\, \mathrm{SDP}_n(A).$$

Applying the rounding scheme to $u_1, \dots, u_m \in S^{n-1}$ gives $x_1, \dots, x_m \in \{-1, +1\}$ with

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^m \sum_{j=1}^m A_{ij}\ x_i x_j\right] &= \frac{2}{\pi}\sum_{i=1}^m \sum_{j=1}^m A_{ij} \arcsin u_i \cdot u_j \\ &\geq \frac{2\rho}{\pi\gamma(n)}\,\mathrm{SDP}_n(A),\end{aligned}$$

where we used that the matrix A and the matrix

$$\left(\frac{2}{\pi}\left(\arcsin u_i \cdot u_j - \frac{u_i \cdot u_j}{\gamma(n)}\right)\right)_{1\leq i,j\leq m}$$

are both positive semidefinite. The last statement follows from Lemma 1 applied to the vectors $u_1, \dots, u_m$ lying in S^{n-1}. Since $\mathrm{SDP}_n(A) \geq \mathrm{SDP}_1(A)$, this is a polynomial-time approximation algorithm for SDP_1 with approximation ratio at least $(2\rho)/(\pi\gamma(n))$. The UGC hardness result of Khot and Naor now implies that $\rho \leq \gamma(n)$. □

6 The Case of Laplacian Matrices

In this section we show that one can improve the approximation ratio of the algorithm if the positive semidefinite matrix $A = (A_{ij}) \in \mathbb{R}^{m\times m}$ has the following special structure:

$$A_{ij} \le 0, \quad \text{if } i \ne j,$$
$$\sum_{i=1}^{n} A_{ij} = 0, \quad \text{for every } j = 1, \dots, n.$$

This happens for instance when A is the Laplacian matrix of a weighted graph with nonnegative edge weights. A by now standard argument due to Goemans and Williamson [6] shows that the algorithm has the approximation ratio

$$v(n) = \min \left\{ \frac{1 - E_n(t)}{1 - t} : t \in [-1, 1] \right\}.$$

To see this, we write out the expected value of the approximation and use the properties of A:

$$\begin{aligned}
\mathbb{E}\Big[\sum_{i,j=1}^{n} A_{ij} x_i \cdot x_j \Big] &= \sum_{i,j=1}^{n} A_{ij} E_n(u_i \cdot u_j) \\
&= \sum_{i \ne j} (-A_{ij}) \left(\frac{1 - E_k(u_i \cdot u_j)}{1 - u_i \cdot u_j} \right) (1 - u_i \cdot u_j) \\
&\ge v(n) \sum_{i,j=1}^{n} A_{ij} u_i \cdot u_j \\
&= v(n)\, \mathrm{SDP}_\infty(A).
\end{aligned}$$

The case $n = 1$ corresponds to the MAX CUT approximation algorithm of Goemans and Williamson [6]. For this we have

$$v(1) = 0.8785\ldots, \quad \text{minimum attained at } t_0 = -0.689\ldots.$$

We computed the values $v(2)$ and $v(3)$ numerically and got

$$v(2) = 0.9349\ldots, \quad \text{minimum attained at } t_0 = -0.617\ldots,$$
$$v(3) = 0.9563\ldots, \quad \text{minimum attained at } t_0 = -0.584\ldots.$$

Acknowledgements

We thank Joe Eaton, Monique Laurent, Robb Muirhead, and Achill Schürmann for helpful discussions. The third auhor thanks the Institute for Pure & Applied Mathematics at UCLA for its hospitality and support.

References

1. Andrews, G.E., Askey, R., Roy, R.: Special functions. Cambridge University Press, Cambridge (1999)
2. Alon, N., Makarychev, K., Makarychev, Y., Naor, A.: Quadratic forms on graphs. Invent. Math. 163, 499–522 (2006)

3. Alon, N., Naor, A.: Approximating the cut-norm via Grothendieck's inequality. SIAM J. Comp. 35, 787–803 (2006)
4. Ben-Tal, A., Nemirovski, A.: On tractable approximations of uncertain linear matrix inequalities affected by interval uncertainty. SIAM J. Optim. 12, 811–833 (2002)
5. Briët, J., Buhrman, H., Toner, B.: A generalized Grothendieck inequality and entanglement in XOR games (January 2009) (preprint)
6. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM 42, 1115–1145 (1995)
7. Grothendieck, A.: Résumé de la théorie métrique des produits tensoriels topologiques. Bol. Soc. Mat. São Paulo 8, 1–79 (1953)
8. Jameson, G.J.O.: Summing and nuclear norms in Banach space theory. Cambridge University Press, Cambridge (1987)
9. Karakostas, G.: A better approximation ratio for the vertex cover problem. In: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, pp. 1043–1050. Springer, Heidelberg (2005)
10. Khot, S.: On the power of unique 2-prover 1-round games. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, pp. 767–775 (2002)
11. Khot, S., Kindler, G., Mossel, E., O'Donnell, R.: Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? SIAM J. Comput. 37, 319–357 (2007)
12. Khot, S., Naor, A.: Approximate kernel clustering. In: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 561–570. IEEE Computer Society, Los Alamitos (2008)
13. Khot, S., Naor, A.: Sharp kernel clustering algorithms and their associated Grothendieck inequalities, preprint (June 2009)
14. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2-\varepsilon$. In: Proceedings of the 18th Annual IEEE Conference on Computational Complexity, pp. 379–386. IEEE Computer Society, Los Alamitos (2003)
15. Linial, N., Shraibman, A.: Lower bounds in communication complexity based on factorization norms. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, pp. 699–708 (2007)
16. Mahajan, S., Ramesh, H.: Derandomizing approximation algorithms based on semidefinite programming
17. Muirhead, R.J.: Aspects of multivariate statistical theory. John Wiley & Sons, Chichester (1982)
18. Nesterov, Y.E.: Semidefinite relaxation and nonconvex quadratic optimization. Optimization Methods and Software 9, 141–160 (1998)
19. Raghavendra, P.: Optimal algorithms and inapproximability results for every csp? In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 245–254 (2008)
20. Raghavendra, P., Steurer, D.: Towards computing the Grothendieck constant. In: ACM-SIAM Symposium on Discrete Algorithms, pp. 525–534 (2009)
21. Rietz, R.E.: A proof of the Grothendieck inequality. Israel J. Math. 19, 271–276 (1974)
22. Schoenberg, I.J.: Positive definite functions on spheres. Duke Math. J. 9, 96–108 (1942)

Cycle Detection and Correction

Amihood Amir[1,2,⋆], Estrella Eisenberg[1,⋆⋆], Avivit Levy[3,4], Ely Porat[1], and Natalie Shapira[1]

[1] Department of Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel
{amir,porately,davidan1}@cs.biu.ac.il
[2] Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218
[3] Department of Software Engineering, Shenkar College,
12 Anna Frank, Ramat-Gan, Israel
avivitlevy@shenkar.ac.il
[4] CRI, Haifa University, Mount Carmel, Haifa 31905, Israel

Abstract. Assume that a natural cyclic phenomenon has been measured, but the data is corrupted by errors. The type of corruption is application-dependent and may be caused by measurements errors, or natural features of the phenomenon. This paper studies the problem of recovering the correct cycle from data corrupted by various error models, formally defined as the *period recovery problem*. Specifically, we define a metric property which we call *pseudo-locality* and study the period recovery problem under pseudo-local metrics. Examples of pseudo-local metrics are the Hamming distance, the swap distance, and the interchange (or Cayley) distance. We show that for pseudo-local metrics, periodicity is a powerful property allowing *detecting* the original cycle and *correcting* the data, under suitable conditions. Some surprising features of our algorithm are that we can *efficiently* identify the period in the corrupted data, up to a number of possibilities logarithmic in the length of the data string, even for metrics whose calculation is $\mathcal{NP}$*-hard*. For the Hamming metric we can reconstruct the corrupted data in near linear time even for unbounded alphabets. This result is achieved using the property of *separation* in the self-convolution vector and Reed-Solomon codes. Finally, we employ our techniques beyond the scope of pseudo-local metrics and give a recovery algorithm for the non pseudo-local Levenshtein edit metric.

1 Introduction

Cyclic phenomena are ubiquitous in nature, from Astronomy, Geology, Earth Science, Oceanography, and Meteorology, to Biological Systems, the Genome, Economics, and more. Part of the scientific process is understanding and explaining these phenomena. The first step is identifying these cyclic occurrences.

Assume, then, that an instrument is making measurements at fixed intervals. When the stream of measurements is analyzed, it is necessary to decide whether these measurements represent a cyclic phenomenon. The "cleanest" version of

⋆ Partially supported by NSF grant CCR-09-04581 and ISF grant 347/09.
⋆⋆ Supported by a Bar-Ilan University President Fellowship.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 43–54, 2010.

this question is whether the string of measurements is *periodic*. Periodicity is one of the most important properties of a string and plays a key role in data analysis. As such, it has been extensively studied over the years [16] and linear time algorithms for exploring the periodic nature of a string were suggested (e.g. [12]). Multidimensional periodicity [4,14,18] and periodicity in parameterized strings [6] was also explored.

However, realistic data may contain errors. Such errors may be caused by the process of gathering the data which might be prone to transient errors. Moreover, errors can also be an inherent part of the data because the periodic nature of the data represented by the string may be inexact. Nevertheless, it is still valuable to detect and utilize the underlying cycle. Assume, then, that, in reality, there is an underlying periodic string, which had been corrupted. Our task is to discover the original uncorrupted string.

This seems like an impossible task. To our knowledge, reconstruction or correction of data is generally not possible from raw natural data. The field of Error Correcting Codes is based on the premise that the original data is not the transmitted data. Rather, it is converted to another type of data with features that allow correction under the appropriate assumptions. Without this conversion, errors on the raw data may render it totally uncorrectable. A simple example is the following: consider the string *aaaaa aaaaa aaaaa aaaab*. This may be the string *aaaaa aaaaa aaaaa aaaaa* with one error at the end (an a was replaced by a b), or the string *aaaaa aaaab aaaaa aaaab* with one error at the 10th symbol (a b was replaced by an a). How can one tell which error it was?

In this paper we show that, surprisingly, data periodicity acts as a feature to aid the data correction under various error models. The simplest natural error model is substitution errors. It generally models the case where the errors may be transient errors due to transmission noise or equipment insensitivity.

Of course, too many errors can completely change the data, making it impossible to identify the original data and reconstruct the original cycle. On the other hand, it is intuitive that few errors should still preserve the periodic nature of the original string. The scientific process assumes a great amount of confidence in the measurements of natural phenomena, otherwise most advances in the natural sciences are meaningless. Thus, it is natural to assume that the measurements we receive are, by and large, accurate. Therefore, it is reasonable to assume that we are presented with data that is faithful to the original without too many corruptions. Formally, the problem is defined as follows:

The Period Recovery Problem. Let S be n-long string with period P. Given S', which is S possibly corrupted by at most k errors under a metric d, return P.

The term "recovering" is, in a sense, approximating the original period, because it may be impossible to distinguish the original period from other false candidates. The "approximation" of the original period means identifying a small set of candidates that is guaranteed to include the original period. We are able to provide such a set of size $O(\log n)$.

We quantify the number of errors k that still guarantees the possibility of such a recovery. It turns out that this number is dependent on the size of the original cycle. For example, if the cycle is of size 2, up to 12% substitution errors **in any distribution** can be tolerated to enable almost linear time recovery. The number of errors that still allow correction of the data is quantified as a function of the period length *in the worst case*.

We take a further step and consider more general error models. Such metrics may model specific types of natural corruptions that are application dependent. Examples of such are *reversals* [9], *transpositions* [8], *block-interchanges* [11], *interchanges* [10,2], or *swaps* [3]. We study the problem of recovering the cyclic phenomenon for a general set of metrics that satisfy a condition, which we call *pseudo-locality*. Intuitively, pseudo-locality means that a single corruption has, in some sense, a local effect on the number of mismatches. The set of pseudo-local metrics is quite extensive and includes such well-studied metrics as the *Hamming Distance*, *Swap Distance*, and *Interchange (*or *Cayley) Distance*.

Results. We give a bound on the number of errors that still allow detection of $O(\log n)$ candidates for the original period, where n is the length of the measured raw data under **any pseudo-local metric**. The original underlying cycle is guaranteed to be one of these candidates. To our surprise, this recovery can be done efficiently **even for metrics whose computation is $\mathcal{NP}$-hard**. We also give faster (near linear time) recovery algorithm for the Hamming distance. Finally, we show that our techniques can be employed even beyond the scope of pseudo-local metrics and give a recovery algorithm for the edit distance, which is not a pseudo-local metric. Let S be n-long string with period P of length p. We prove the following:

• Let d be a c-pseudo-local-metric and let $f_d(n)$ be the complexity of computing the distance between two n-long strings under the metric d. Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors under the metric d, then, a set of $\log_{1+\frac{\varepsilon}{c}} n$ candidates which includes P can be constructed in time $O(n \log^2 n + f_d(n) \cdot n \log n)$ (Theorem 1).

As a corollary we get that if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ swap errors, then, a set of $\log_{1+\frac{\varepsilon}{2}} n$ candidates which includes P can be constructed in time $O(n^2 \log^4 n)$ (Corollary 3).

• Let d be a c-pseudo-local metric that admits a polynomial time γ-approximation algorithm with complexity $f_d^{app}(n)$, where $\gamma > 1$ is a constant. Let $\varepsilon > (\gamma - 1) \cdot 2c$ be a constant. Then, if S is corrupted by at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors under the metric d, then, a set of $\log_{1+\frac{\varepsilon'}{c}} n$ candidates which includes P can be constructed in time $O(n \log^2 n + f_d^{app}(n) \cdot n \log n)$, where $\varepsilon' = \frac{\varepsilon}{\gamma} - \frac{2c(1-\gamma)}{\gamma} > 0$ is a constant. (Theorem 2)

As a corollary we get that for every $\varepsilon > \frac{8}{9}$, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ interchange errors, then, a set of $\log_{1+\frac{\varepsilon'}{2}} n$ candidates which includes P can be constructed in time $O(n^2 \log^2 n)$, where $\varepsilon' = \frac{3\varepsilon}{2} - \frac{4}{3} > 0$ is a constant. (Corollary 4)

Note that the period can be approximated in *polynomial time*, even though computing the interchange distance is $\mathcal{NP}$*-hard* [5].

- If S is corrupted by less than $\frac{n}{4p}$ substitution errors then a set of $O(\log n)$ candidates that includes P can be constructed in time $O(n \log n)$ for bounded alphabets (Theorem 3),
and $O(n \log^2 n)$ for unbounded alphabets (Theorem 4).
- Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ Levenshtein edit operations, then, a set of $\log_{1+\frac{\varepsilon}{2}} n$ candidates such that their cyclic rotations include P, can be constructed in time $O(n^3 \log n)$. (Theorem 5)

Techniques. New concepts were defined, studied and used in this paper in order to achieve the results. The first important concept is the pseudo-locality property. Identifying and defining this property enabled a unified study of cycle recovery under a rich set of metrics that includes well-known metrics as the *Hamming Distance*, *Swap Distance*, and *Interchange Distance*. Properties of pseudo-local metrics were proved to support a non-trivial use of the Minimum Augmented Suffix Tree data structure in order to achieve our general results.

Another useful concept defined in this paper is *separation* in the *self-convolution vector*. This property is used as a basic tool to aid in faster discovery of the original period under the Hamming distance. Separation is a seemingly very strong property. However, we prove sufficient conditions under which separation can be achieved. We extend our results to unbounded alphabets using *Reed-Solomon* codes, achieving the same results with a degradation of a $\log n$ multiplicative factor. It is somewhat surprising that a tool of Error Correction is brought to bear on achieving correction in raw natural data. The key to this success is the periodicity property. We stress that in Error-Correcting-Codes the pre-designed structure of the code is known to the decoder. In our situation, however, we attempt to find a cyclic phenomenon where nothing is known a-priori, not even if it really exists.

2 Preliminaries

In this section we give basic definitions of periodicity and approximate periodicity, string metrics and pseudo-local metrics. We also give some basic lemmas.

Definition 1. *Let S be a string of length n. S is called* periodic *if $S = P^i pref(P)$, where $i \in \mathbb{N}$, $i \geq 2$, P is a substring of S such that $|P| \leq n/2$, P^i is the concatenation of P to itself i times, and $pref(P)$ is a prefix of P. The smallest such substring P is called the* period *of S. If S is not periodic it is called* aperiodic.[1]

Remark. Throughout the paper we use p to denote a period length and P the period string, i.e., $p = |P|$.

Definition 2. *Let S be n-long string over alphabet Σ. Let d be a metric defined on strings. S is called* periodic with k errors *if there exists a string P over Σ,*

[1] Denote by $|S|$ the length of a string S.

$p \in \mathbb{N}$, $p \leq n/2$, *such that* $d(P^{\lfloor n/p \rfloor} pref(P), S) = k$, *where* $pref(P)$ *is a prefix of* P *of length* $n \bmod p$. *The string* P *is called* a k-error period of S *or* approximate period of S with k errors.

Consider a set Σ and let x and y be two n-long strings over Σ. We wish to formally define the process of converting x to y through a sequence of operations. An *operator* ψ is a function $\psi : \Sigma^n \rightarrow \Sigma^{n'}$, with the intuitive meaning being that ψ converts n-long string x to n'-long string y with a cost associated to ψ. That cost is the distance between x and y. Formally,

Definition 3 [string metric]. *Let* $s = (\psi_1, \psi_2, \ldots, \psi_k)$ *be a sequence of operators, and let* $\psi_s = \psi_1 \circ \psi_2 \circ \cdots \circ \psi_k$ *be the composition of the* ψ_j*'s. We say that* s converts x into y *if* $y = \psi_s(x)$.
Let Ψ *be a set of rearrangement operators, we say that* Ψ can convert x to y, *if there exists a sequence* s *of operators from* Ψ *that converts* x *to* y. *Given a set* Ψ *of operators, we associate a non-negative* cost *with each sequence from* Ψ, $cost : \Psi^* \rightarrow R^+$. *We call the pair* $(\Psi, cost)$ *an* edit system. *Given two strings* $x, y \in \Sigma^*$ *and an edit system* $\mathcal{R} = (\Psi, cost)$, *we define the distance from* x *to* y *under* $\mathcal{R}$ *to be:* $d_{\mathcal{R}}(x, y) = \min\{cost(s) | s \text{ from } \mathcal{R} \text{ converts } x \text{ to } y\}$. *If there is no sequence that converts* x *to* y *then the distance is* ∞.

It is easy to verify that $d_{\mathcal{R}}(x, y)$ is a metric. Definition 4 gives examples of string metrics.

Definition 4. • Hamming distance: $\Psi = \{\rho^n_{i,\sigma} | i, n \in \mathbb{N}, \quad i \leq n, \quad \sigma \in \Sigma\}$, *where* $\rho^n_{i,\sigma}(\alpha)$ *substitutes the* i*th element of* n*-tuple* α *by symbol* σ. *We denote the Hamming distance by* H.

• Edit distance: *In addition to the substitution operators of the Hamming distance,* Ψ *also has* insertion *and* deletion *operators. The insertion operators are:* $\{\iota^n_{i,\sigma} | i, n \in \mathbb{N}, \quad i \leq n, \quad \sigma \in \Sigma\}$, *where* $\iota^n_{i,\sigma}(\alpha)$ *adds the symbol* σ *following the* i*th element of* n*-tuple* α, *creating an* $n+1$*-tuple* α'.
The deletion operators are $\{\delta^n_i | i, n \in \mathbb{N}, \quad i \leq n\}$, *where* $\delta^n_i(\alpha)$ *deletes the symbol at location* i *of* n*-tuple* α, *creating an* $n-1$*-tuple* α'.

• Swap distance: $\Psi = \{\zeta^n_i | i, n \in \mathbb{N}, \quad i < n\}$, *where* $\zeta^n_i(\alpha)$ *swaps the* i*th and* $i+1$*st elements of* n*-tuple* α, *creating an* n*-tuple* α'. *A* valid *sequence of operators in the Swap metric has the additional condition that if* ζ^n_i *and* ζ^m_j *are operators in a sequence then* $i \neq j$, $i \neq j+1$, $i \neq j-1$, *and* $n = m$.

• Interchange distance: $\Psi = \{\pi^n_{i,j} | i, n \in \mathbb{N}, \quad i \leq j \leq n\}$, *where* $\pi^n_{i,j}(\alpha)$ *interchanges the* i*th and* j*th elements of* n*-tuple* α, *creating an* n*-tuple* α'.

Definition 5 [pseudo-local metric]. *Let* d *be a string metric.* d *is called a* pseudo-local metric *if there exists a constant* $c \geq 1$ *such that, for every two strings* S_1, S_2, *if* $d(S_1, S_2) = 1$ *then* $1 \leq H(S_1, S_2) \leq c$.
A metric that is pseudo-local with constant c *is called a* c-pseudo-local metric.

Note that pseudo-locality allows the resulted number of mismatches to be unboundedly far from each other (as may happen in an interchange) and therefore, a pseudo-local metric is not necessarily also local in the intuitive sense. Lemma 1 follows immediately from Definitions 4 and 5.

Lemma 1. *The following metrics are c-pseudo-local metrics: Hamming distance (with $c = 1$), Swap distance (with $c = 2$), Interchange distance (with $c = 2$).*

On the other hand, the Edit distance is not a pseudo-local metric, because a single deletion or insertion may cause an unbounded number of mismatches. Lemma 2 states a basic property of pseudo-local metrics exploited in this paper.

Lemma 2. *Let d be a c-pseudo-local metric. Then, for every $n \in \mathbb{N}$ and n-long periodic strings S_1, S_2 with P_1 and P_2 the periods of S_1 and S_2 respectively, $p_1 \geq p_2$ and $P_1 \neq P_2$ (i.e., there exists at least one mismatch between P_1 and P_2) it holds that $d(S_1, S_2) \geq \frac{n}{c \cdot p_1}$.*

3 Approximating the Period under Pseudo-Local Metrics

The term *approximation* here refer to the ability to give a relatively small size set of candidates that *includes* the *exact* original period of the string. We first, show the bounds on the number of errors that still guarantees a small-size set of candidates. We then show how this set can be identified, and thus the original uncorrupted string can be *approximately* recovered.

Lemma 3. *Let d be a c-pseudo-local metric. Let $\varepsilon > 0$ be a constant, S an n-long string and P_1, P_2, $P_1 \neq P_2$, approximate periods of S with at most $\frac{n}{(2c+\varepsilon)\cdot p_1}$, $\frac{n}{(2c+\varepsilon)\cdot p_2}$ errors respectively (w.l.o.g. assume that $p_1 \geq p_2$). Then, $p_1 \geq (1+\frac{\varepsilon}{c})\cdot p_2$.*

Corollary 1. *Let d be a c-pseudo-local-metric. Let S be a n-long string. Then, there are at most $\log_{1+\frac{\varepsilon}{c}} n$ different approximate periods P of S with at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors.*

The General Period Recovery Algorithm. We now describe an algorithm for recovering the original period up to $O(\log n)$ candidates under any pseudo-local metric defining the string corruption process. The algorithm has two stages: candidates extraction stage and validation stage. The algorithm starts by extracting from the string S a set of candidates for the original period. Due to pseudo-locality of the metric and the number of errors assumed, there are copies of the original period that are left uncorrupted . A trivial $O(n^2)$-size set would, therefore, be the set of all substrings of S.

We, however, do better than that. Our algorithm extracts a set of only $O(n \log n)$ candidates. These candidates are then verified to find those that comply with the given error bound. By Corollary 1, we know that only $O(\log n)$ candidates can survive the validation stage.

The algorithm extraction stage uses the $MAST$ (Minimal Augmented Suffix Tree) data structure presented by [7]. This data structure is a suffix-tree with additional nodes and information in each node. The additional information in a node is the number of non-overlapping occurrences of the string represented by the path from the root to that node. The number of nodes in the $MAST$ is proven in [7] to be $O(n \log n)$, and this is also its space bound. The $MAST$

construction time is $O(n \log^2 n)$ time. As we show in Lemma 4, from the $MAST$ nodes, a set of candidates that includes all possible approximate periods can be extracted. To do that, the algorithm performs an *Extended Depth First Search* on $MAST(S)$, denoted *EDFS*, which also keeps track of the string representing the path from the root to the current node. We use the following notations for the EDFS on $MAST(S)$. $S(v)$ is the string associated with the path from the root of $MAST(S)$ to node v. It is important to note that the strings associated with paths in $MAST(S)$ are only implicitly kept, as references to the corresponding locations in S (as it is in a Suffix Tree). This enables the $MAST$ data structure to consume only near linear space. In the validation stage, for each candidate P the distance between S_P and S is checked if it is at most $\frac{n}{(2c+\varepsilon)\cdot p}$.

Lemma 4. *Let d be a c-pseudo-local metric. Let $\varepsilon > 0$ be a constant, S n-long string with P an approximate period of S with at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors. Then, if $p > \frac{c}{\varepsilon}$, there exists a node v in $MAST(S)$ such that $S(v) = P$ or $S(v)$ is the prefix of P of length $p-1$.*

Theorem 1. *Let d be a c-pseudo-local-metric and let $f_d(n)$ be the complexity of computing the distance between two n-long strings under the metric d. Let S be n-long string with period P. Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors under the metric d, then, a set of $\log_{1+\frac{\varepsilon}{c}} n$ candidates which includes P can be constructed in time $O(n \log^2 n + f_d(n) \cdot n \log n)$.*

Proof. Let S' be the given corruption of S. It is enough to show that the general period recovering algorithm on input S' has the requested property and its time is bounded by $O(n \log^2 n + f_d(n) \cdot n \log n)$. We first show that every approximate period P' of S' with at most $\frac{n}{(2c+\varepsilon)\cdot p'}$ errors is put in C in the extraction stage of the algorithm (and thus P is put in this set). By Lemma 4, P' falls into one of the following three cases: (1) $p' \leq \frac{c}{\varepsilon}$, (2) $P' = S(v)$ for some node v in $MAST(S')$, and (3) $P' = s_e$ for some edge e in $MAST(S')$. In each of these cases, P' is extracted by the algorithm in the extraction phase. Since, P' is an approximate period of S' with at most $\frac{n}{(2c+\varepsilon)\cdot p'}$ errors, it also survives the algorithm's validation stage. Therefore, the set of candidates returned by the algorithm includes P. By Corollary 1 we have that the size of the set returned by the algorithm is $\log_{1+\frac{\varepsilon}{c}} n$, as required.

We now prove the time bound of the algorithm. [7] assure that building $MAST(S')$ can be done in time $O(n \log^2 n)$. Since $\frac{c}{\varepsilon}$ is constant, extraction of substrings within case (1) can be done in $O(n)$ time. The space of the $MAST(S')$ constructed is $O(n \log n)$ and its nodes number is also $O(n \log n)$ [7]. Therefore, the EDFS performed takes time linear in the $MAST(S')$ size, i.e., $O(n \log^2 n)$ time. Also, the size of the extracted candidates set before the validation is $O(n \log n)$, as explained above. Only for these $O(n \log n)$ candidates the validation stage is done. The theorem then follows. □

From Theorem 1 we get a simple cycle recovery under the Hamming distance (Corollary 2), recovery under the swap distance (Corollary 3) and recovery

under any pseudo-local metric that admits efficient constant approximation (Theorem 2), specifically, under the interchange distance (Corollary 4).

Corollary 2. *Let S be n-long string with period P. Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(2+\varepsilon)\cdot p}$ substitution errors, then, a set of $\log_{1+\varepsilon} n$ candidates which includes P can be constructed in time $O(n^2 \log n)$.*

Corollary 3. *Let S be n-long string with period P. Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ swap errors, then, a set of $\log_{1+\frac{\varepsilon}{2}} n$ candidates which includes P can be constructed in time $O(n^2 \log^4 n)$.*

Theorem 2. *Let S be n-long string with period P. Let d be a c-pseudo-local metric that admits a polynomial time γ-approximation algorithm with complexity $f_d^{app}(n)$, where $\gamma > 1$ is a constant. Let $\varepsilon > (\gamma - 1) \cdot 2c$ be a constant. Then, if S is corrupted by at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors under the metric d, then, a set of $\log_{1+\frac{\varepsilon'}{c}} n$ candidates which includes P can be constructed in time $O(n \log^2 n + f_d^{app}(n) \cdot n \log n)$, where $\varepsilon' = \frac{\varepsilon}{\gamma} - \frac{2c(1-\gamma)}{\gamma} > 0$ is a constant.*

Corollary 4. *Let S n-long string with period P. Let $\varepsilon > \frac{8}{9}$ be a constant. Then, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ interchange errors, then, a set of $\log_{1+\frac{\varepsilon'}{2}} n$ candidates which includes P can be constructed in time $O(n^2 \log^2 n)$, where $\varepsilon' = \frac{3\varepsilon}{2} - \frac{4}{3} > 0$ is a constant.*

4 Faster Period Recovery under the Hamming Distance

By Corollary 2, the general algorithm describes in Sect. 3 gives a $O(n^2 \log n)$ time bound for the Hamming metric. In this section we use stronger tools to achieve cycle recovery under the Hamming metric in near linear time. We need some metric-specific definitions and tools.

Definition 6. *Let S be n-long string over alphabet Σ, and let $p \in \mathbb{N}$, $p \leq n/2$. For every i, $0 \leq i \leq p-1$, the i-th frequent element with regard to p, denoted F_i^p, is the symbol $\sigma \in \Sigma$ which appears the largest number of times in locations $i + c \cdot p$ in S, where $c \in [0..\lfloor n/p \rfloor]$, and[2] $i + c \cdot p \leq n$.*

The Self-Convolution Vector. The main tool we exploit in this paper to give faster recovery for the Hamming metric is the *self-convolution vector* defined below. Using standard FFT techniques gives Lemma 5.

[2] For simplicity of exposition, positive integers of the form $i + c \cdot p$, $c \in [0..\lfloor n/p \rfloor]$ for some $p \leq n/2$ are referred to as locations in S without mentioning explicitly the condition $i + c \cdot p \leq n$. It is always assumed that only valid positions in S are referred to.

Definition 7. *Let S be n-long string over alphabet Σ, and let $\bar{S}$ be the string S concatenated with n \$'s (where \$ $\notin \Sigma$). The* self-convolution vector of S, v, *is defined for every i, $0 \leq i \leq n/2-1$, $v[i] = \sum_{j=0}^{n-1} f(\bar{S}[i+j], S[j])$, where* $f(\bar{S}[i+j], S[j]) = \begin{cases} 1, \text{ if } \bar{S}[i+j] \neq S[j] \text{ and } \bar{S}[i+j] \neq \$; \\ 0, \text{ otherwise.} \end{cases}$.

Lemma 5. [13] *The self-convolution vector of a length n string S over alphabet Σ can be computed in time $O(|\Sigma| n \log n)$.*

The self-convolution vector of a periodic string (without corruptions) is a highly structured vector. Lemma 6 describes this structure.

Lemma 6. *Let S be n-long string with period P, then the self-convolution vector of S is 0, in every location i, $i \equiv 0 \bmod p$, and at least $\lfloor \frac{n}{p} \rfloor$, in all other locations.*

The Separation Property. The structure of the self-convolution vector in an uncorrupted string S is described by Lemma 6. By this structure the locations that are multiplicities of the period can be easily identified, and therefore also the length of the period can be easily found. The separation property of the self-convolution vector of a given possibly corrupted string S' relaxes the demand for 0 at the 0 mod p locations and at least $\lfloor \frac{n}{p} \rfloor$ elsewhere, but insists on a fixed separation nevertheless.

Definition 8. *Let S' be n-long string with period P, and let v be the self-convolution vector of S'. We say that v is* (α, β)-separable *if $0 \leq \alpha < \beta$, and v has values:*
$< \alpha \frac{n}{p}$, in every location i, $i \equiv 0$ mod p, and
$> \beta \frac{n}{p}$, in all other locations.

Separation is a seemingly very strong property. Does it ever occur in reality? Lemma 7 shows that if there are less substitution errors than $\frac{n}{4p}$, separation is guaranteed.

Lemma 7. *Let S be n-long string with period P. Let S' be the string S corrupted with at most $\lfloor \delta \frac{n}{p} \rfloor$ substitution errors, where $0 \leq \delta < \frac{1}{4}$, then the self-convolution vector of S' is $(2\delta, 1-2\delta)$-separable.*

The Period *H*-Recovery Algorithm. The period recovery algorithm under Hamming metric first gathers information on the corrupted string S' by computing the self-convolution vector v. Then, (α, β)-separation is used in order to reconstruct the original period. By Lemma 7, if there are not too many substitution errors, separation is guaranteed. The algorithm sorts all values of the self-convolution vector in ascending order $v_{i_1}, ..., v_{i_n}$ and considers all locations where $\frac{v_{i_{\ell+1}}}{v_{i_\ell}} \geq \frac{\beta}{\alpha}$. Such an i_ℓ is called a *separating location*, and v_{i_ℓ}, a *separating value*. Clearly there are no more that $O(\log n)$ separating locations. Each one needs to be verified if it indeed defines a period. The algorithm uses the fact that

all the multiplicities of the original period have values at most the separating value, in order to find the period length, which is the common difference between the sorted list of multiplicities. Given the period length, the period can be easily reconstructed by a linear scan and the majority criterion.

We now give the worst-case analysis of Algorithm Period H-Recovery. Lemma 8 proves the algorithm's correctness. The complexity guarantee of the algorithm is given by Lemma 9. Theorem 3 follows.

Lemma 8. *Let S be n-long string with period P, and let S' be S corrupted by substitution errors. Assume that the self-convolution vector of S' is (α, β)-separable. Then algorithm Period H-Recovery returns a non empty set $\hat{C}$, $|\hat{C}| = O(\log n)$, such that $P \in \hat{C}$.*

Lemma 9. *Algorithm Period H-Recovery runs in $O(|\Sigma| n \log n)$ steps.*

Theorem 3. *Let S be n-long string with period P. Then, if S is corrupted but preserves a (α, β) separation, a set of $O(\log n)$ candidates which includes P can be constructed in time $O(|\Sigma| n \log n)$.*

Unbounded Alphabets. Theorem 3 assures that we can give a set of $O(\log n)$ candidates which includes P, for a corrupted n-length string S with period P that preserves a (α, β) separation, in time $O(|\Sigma| n \log n)$. For small alphabets this result is fine, but for unbounded alphabets, e.g. of size $p + n/p$, it is inefficient. Of course, one can immediately lower the time complexity to $O(n\sqrt{p \log n})$ by using Abrahamson's result [1]. But we can do even better by using Reed-Solomon codes [17] to encode the alphabet $\{1, ..., n\}$.

Theorem 4. *Let S be n-long string with period P over alphabet Σ. Then, if S is corrupted but preserves a (α, β) separation, a set of $O(\log n)$ candidates which includes P can be constructed in time $O(n \log^2 n)$.*

5 Period Recovery under the Edit Distance

As mentioned previously, the edit distance is not pseudo-local. In fact, it is also clear that Corollary 1 does not hold. As an example consider a period P of length $n^{1/4}$. The allotted number of corruptions is $\frac{n}{(2+\varepsilon)n^{1/4}} = \frac{n^{3/4}}{(2+\varepsilon)}$. Nevertheless, every one of the $n^{1/4}$ cyclic rotations of P generates a string whose edit distance from $P^{3/4}$ is at most $n^{1/4}$, by an appropriate number of leading deletions. This number of errors is well within the bounds tolerated by Corollary 1, yet the number of possible approximate periods is quite large. Fortunately, Lemma 10 assures that even for edit distance, there are $O(\log n)$ candidates *up to cyclic rotations*.

Lemma 10. *Let $\varepsilon > 0$ be a constant. Let S an n-long string and let P_1, P_2, $P_1 \neq P_2$, be approximate periods of S with at most $\frac{n}{(4+\varepsilon)\cdot p_1}$, $\frac{n}{(4+\varepsilon)\cdot p_2}$ edit errors respectively (w.l.o.g. assume that $p_1 \geq p_2$), such that P_1 is not a cyclic rotation of P_2. Then, $p_1 \geq (1 + \frac{\varepsilon}{2}) \cdot p_2$.*

Corollary 5. *Let S be a n-long string. Then, there are at most $\log_{1+\frac{\varepsilon}{2}} n$ different approximate periods P of S with at most $\frac{n}{(4+\varepsilon)\cdot p}$ edit errors, that are not cyclic rotations of any other approximate period.*

The naive construction of the approximate period candidates is done as follows. For each of the $O(n \log n)$ candidates output by the extraction stage of the general Period Recovery Algorithm, and each of its rotations (for a total of $O(n^2 \log n)$ candidates), compute the edit distance with the input string. The edit distance is computable in time $O(n^2)$ [15] by dynamic programming. Thus, the total time is $O(n^4 \log n)$. But we can do better by using different methods for handling small/large candidates (of length at most/ exceeds $\sqrt{n}$, respectively) and combining with recent results in computation of ED [19].

Theorem 5. *Let S be n-long string with period P. Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ Levenshtein edit operations, then, a set of $\log_{1+\frac{\varepsilon}{2}} n$ candidates such that their cyclic rotations include P, can be constructed in time $O(n^3 \log n)$.*

References

1. Abrahamson, K.: Generalized string matching. SIAM J. Comp. 16(6), 1039–1051 (1987)
2. Amir, A., Aumann, Y., Benson, G., Levy, A., Lipsky, O., Porat, E., Skiena, S., Vishne, U.: Pattern matching with address errors: rearrangement distances. In: Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1221–1229 (2006)
3. Amir, A., Aumann, Y., Landau, G., Lewenstein, M., Lewenstein, N.: Pattern matching with swaps. Journal of Algorithms 37, 247–266 (2000); Preliminary version appeared at FOCS 97
4. Amir, A., Benson, G.: Two-dimensional periodicity and its application. SIAM J. Comp. 27(1), 90–106 (1998)
5. Amir, A., Hartman, T., Kapah, O., Levy, A., Porat, E.: On the cost of interchange rearrangement in strings. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA 2007. LNCS, vol. 4698, pp. 99–110. Springer, Heidelberg (2007)
6. Apostolico, A., Giancarlo, R.: Periodicity and repetitions in parameterized strings. Discrete Appl. Math. 156(9), 1389–1398 (2008)
7. Apostolico, A., Preparata, F.P.: Data structures and algorithms for the string statistics problem. Algorithmica 15(5), 481–494 (1996)
8. Bafna, V., Pevzner, P.A.: Sorting by transpositions. SIAM J. on Discrete Mathematics 11, 221–240 (1998)
9. Berman, P., Hannenhalli, S.: Fast sorting by reversal. In: Hirschberg, D.S., Myers, E.W. (eds.) CPM 1996. LNCS, vol. 1075, pp. 168–185. Springer, Heidelberg (1996)
10. Cayley, A.: Note on the theory of permutations. Philosophical Magazine (34), 527–529 (1849)
11. Christie, D.A.: Sorting by block-interchanges. Information Processing Letters 60, 165–169 (1996)
12. Crochemore, M.: An optimal algorithm for computing the repetitions in a word. Information Processing Letters 12(5), 244–250 (1981)

13. Fischer, M.J., Paterson, M.S.: String matching and other products. In: Karp, R.M. (ed.) Complexity of Computation, SIAM-AMS Proceedings, vol. 7, pp. 113–125 (1974)
14. Galil, Z., Park, K.: Alphabet-independent two-dimensional witness computation. SIAM J. Comp. 25(5), 907–935 (1996)
15. Levenshtein, V.I.: Binary codes capable of correcting, deletions, insertions and reversals. Soviet Phys. Dokl. 10, 707–710 (1966)
16. Lothaire, M.: Combinatorics on words. Addison-Wesley, Reading (1983)
17. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. SIAM J. Applied Mathematics 8(2), 300–304 (1960)
18. Régnier, M., Rostami, L.: A unifying look at d-dimensional periodicities and space coverings. In: Proc. 4th Symp. on Combinatorial Pattern Matching, vol. 15, pp. 215–227 (1993)
19. Tiskin, A.: Fast distance multiplication of unit-monge matrices. In: Proc. of ACM-SIAM SODA, pp. 1287–1296 (2010)

Decomposition Width of Matroids

Daniel Král'*

Institute for Theoretical Computer Science (ITI)**
Faculty of Mathematics and Physics, Charles University
Malostranské náměstí 25, Prague, Czech Republic
kral@kam.mff.cuni.cz

Abstract. Hliněný [J. Combin. Theory Ser. B 96 (2006), 325–351] showed that every matroid property expressible in the monadic second order logic can be decided in linear time for matroids with bounded branch-width that are represented over finite fields. To be able to extend these algorithmic results to matroids not representable over finite fields, we introduce a new matroid width parameter, the decomposition width, and show that every matroid property expressible in the monadic second order logic can be computed in linear time for matroids given by a decomposition with bounded width. We also relate the decomposition width to matroid branch-width and discuss implications of our results with respect to other known algorithms.

1 Introduction

Algorithmic aspects of graph tree-width form an important part of algorithmic graph theory. Efficient algorithms for computing tree-decompositions of graphs have been designed [1,4] and a lot of NP-complete problems become tractable for classes of graphs with bounded tree-width [2,3]. Most of results of the latter kind are implied by a general result of Courcelle [5,6] that every graph property expressible in the monadic second-order logic can be decided in linear time for graphs with bounded tree-width.

As matroids are combinatorial structures that generalize graphs, it is natural to ask which of these results translate to matroids. Similarly, as in the case of graphs, some hard problems (that cannot be solved in polynomial time for general matroids) can be efficiently solved for (represented) matroids with bounded width. Though the notion of tree-width generalizes to matroids [18,19], a more natural width parameter for matroids is the notion of *branch-width*. Let us postpone its formal definition to Section 2 and just mention at this point that the branch-width of matroids is linearly related with their tree-width, in particular, the branch-width of a graphic matroid is bounded by twice the tree-width of the corresponding graph.

There are two main algorithmic aspects which one needs to address with respect to algorithmic properties of a width parameter of combinatorial structures:

* This research was also partially supported by the GAUK grant 64110.
** Institute for Theoretical Computer Science (ITI) is supported as project 1M0545 by Czech Ministry of Education.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 55–66, 2010.

– the efficiency of computing decompositions with bounded width (if they exist), and
– tractability of hard problems for input structures with bounded width.

The first issue has been successfully settled with respect to matroid branch-width: for every k, there exists a polynomial-time algorithm that either computes a branch-decomposition of an input matroid with width at most k or outputs that there is no such branch-decomposition. The first such algorithm has been found by Oum and Seymour [22] (an approximation algorithm has been known earlier [21], also see [13] for the case of matroids represented over a finite field) and a fixed parameter algorithm for matroids represented over a finite field was later designed by Hliněný and Oum [17].

The tractability results, which include deciding monadic second-order logic properties [11,12,14], computing and evaluating the Tutte polynomial [16] and computing and counting representations over finite fields [20], usually require restricting to matroids represented over finite fields (see Section 2 for the definition). This is consistent with the facts that no subexponential algorithm can decide whether a given matroid is binary [24], i.e., representable over $\mathbb{GF}(2)$, even for matroids with branch-width three and that it is NP-hard to decide representability over $\mathbb{GF}(q)$ for every prime power $q \geq 4$ even for matroids with bounded branch-width represented over $\mathbb{Q}$ [15], as well as with structural results on matroids [7,8,9,10] that also suggest that matroids representable over finite fields are close to graphic matroids (and thus graphs) but general matroids can be quite different.

The aim of this paper is to introduce another width parameter for matroids which will allow to extend the known tractability results to matroids not necessarily representable over finite fields. The cost that needs to be paid for this is that this new width parameter cannot be bounded by any function of the branch-width (as we already mentioned there is no subexponential algorithm to decide whether a matroid with branch-width three is representable over a finite field). On the positive side, the new width parameter is bounded by a function of the branch-width for matroids representable over a fixed finite field and decompositions with bounded width can be computed efficiently (see Section 3). Hence, we are able to unify the already known structural results for matroids. Our new notion captures the "structural finiteness" on cuts represented in the branch decomposition essential for the tractability results and is closely related to rooted configurations as introduced in [7] and indistinguishable sets from [20]. A more logic based approach in this direction has been given in [25].

Our results. In Section 2, we introduce a K-decomposition of a matroid (where K is an integer) and define the decomposition width of a matroid $\mathcal{M}$ to be the smallest integer K such that $\mathcal{M}$ has a K-decomposition. In Section 3, we show that there exists a function $F(k,q)$, the decomposition width of a matroid with branch-width at most k that is representable over $\mathbb{GF}(q)$ is at most $F(k,q)$ and that an $F(k,q)$-decomposition of any such matroid can be computed in polynomial time. In Sections 4, 5 and 6, we show that for every K, there exist polynomial-time algorithms (with the degree of the polynomial independent

of K) for computing and evaluating the Tutte polynomial, deciding monadic second-order logic properties, deciding representability and constructing and counting representations over finite fields when the input matroid is given by its K-decomposition. In particular, our results imply all the tractability results known for matroids represented over finite field (we here claim only the polynomiality, not matching the running times which we did not try to optimize throughout the paper).

A K-decomposition of a matroid actually captures the whole structure of a matroid, i.e., the matroid is fully described by its K-decomposition, and thus it can be understood as an alternative way of providing the input matroid. In fact, for a fixed K, the size of a K-decomposition is linear in the number of matroid elements and thus this representation of matroids is very suitable for this purpose. Let us state (without a proof) that K-decompositions of matroids *support* contraction and deletion of matroid elements without increasing the decomposition width as well as some other matroid operations with increasing the decomposition width by a constant, e.g., relaxing a circuit-hyperplane increases the decomposition width by at most one. By *supporting* we mean that a K-decomposition of the new matroid can be efficiently computed from the K-decomposition of the original one. Hence, the definition of the decomposition width does not only yield a framework extending tractability results for matroids represented over finite fields with bounded branch-width but it also yields a compact data structure for representing input matroids.

2 Matroid Notation

In this section, we formally introduce the notions which are used in this paper. We start with basic notions and we then introduce matroid representations, branch-decompositions and the new width parameter. We also refer the reader to the monographs [23,26] for further exposition on matroids.

A *matroid* $\mathcal{M}$ is a pair $(E, \mathcal{I})$ where $\mathcal{I} \subseteq 2^E$. The elements of E are called *elements* of $\mathcal{M}$, E is the *ground set* of $\mathcal{M}$ and the sets contained in $\mathcal{I}$ are called *independent* sets. The set $\mathcal{I}$ is required to contain the empty set, to be hereditary, i.e., for every $F \in \mathcal{I}$, $\mathcal{I}$ must contain all subsets of F, and to satisfy the exchange axiom: if F and F' are two sets of $\mathcal{I}$ such that $|F| < |F'|$, then there exists $x \in F' \setminus F$ such that $F \cup \{x\} \in \mathcal{I}$. The *rank* of a set F, denoted by $r(F)$, is the size of a largest independent subset of F (it can be inferred from the exchange axiom that all inclusion-wise maximal independent subsets of F have the same size). An important class of matroids form graphic matroids: the ground set of the matroid associated with a graph G are the edges of G and the set of edges is independent if they are acyclic in G. Another example of matroids is formed by uniform matroids $U_{r,n}$ with n elements and independent sets being all subsets with at most r vertices. In the rest, we often understand matroids as sets of elements equipped with a property of "being independent". We use $r(\mathcal{M})$ for the rank of the ground set of $\mathcal{M}$.

If F is a set of elements of $\mathcal{M}$, then $\mathcal{M} \setminus F$ is the matroid obtained from $\mathcal{M}$ by *deleting* the elements of F, i.e., the elements of $\mathcal{M} \setminus F$ are those not contained in F and a subset F' of such elements is independent in the matroid $\mathcal{M} \setminus F$ if and only if F' is independent in $\mathcal{M}$. The matroid $\mathcal{M}/F$ which is obtained by *contraction* of F is the following matroid: the elements of $\mathcal{M}/F$ are those not contained in F and a subset F' of such elements is independent in $\mathcal{M}/F$ if and only if $r(F \cup F') = r(F) + |F'|$ in $\mathcal{M}$. A *loop* of $\mathcal{M}$ is an element e of $\mathcal{M}$ such that $r(\{e\}) = 0$ and a *co-loop* is an element such that $r(\mathcal{M} \setminus \{e\}) = r(\mathcal{M}) - 1$. A *separation* (A, B) is a partition of the elements of $\mathcal{M}$ into two disjoint non-empty sets and a separation is called a *k-separation* if $r(A) + r(B) - r(\mathcal{M}) \leq k - 1$.

2.1 Matroid Representations

Matroids do not generalize only the notion of graphs (recall graphic matroids mentioned earlier) but they also generalize the notion of linear independence of vectors. If $\mathbb{F}$ is a (finite or infinite) field, a mapping $\varphi : E \to \mathbb{F}^d$ from the ground set E of $\mathcal{M}$ to a d-dimensional vector space over $\mathbb{F}$ is a *representation* of $\mathcal{M}$ if a set $\{e_1, \ldots, e_k\}$ of elements of $\mathcal{M}$ is independent in $\mathcal{M}$ if and only if $\varphi(e_1), \ldots, \varphi(e_k)$ are linearly independent vectors in $\mathbb{F}^d$. For a subset F of the elements of $\mathcal{M}$, $\varphi(F)$ denotes the linear subspace of $\mathbb{F}^d$ generated by the images of the elements of F. In particular, $\dim \varphi(F) = r(F)$. Two representations φ_1 and φ_2 of $\mathcal{M}$ are isomorphic if there exists an isomorphism ψ of vector spaces $\varphi_1(E)$ and $\varphi_2(E)$ such that $\psi(\varphi_1(e))$ is a non-zero multiple of $\varphi_2(e)$ for every element e of $\mathcal{M}$.

We next introduce additional notation for vector spaces over a field $\mathbb{F}$. If U_1 and U_2 are two linear subspaces of a vector space over $\mathbb{F}$, $U_1 \cap U_2$ is the linear space formed by all the vectors lying in both U_1 and U_2, and $U_1 + U_2$ is the linear space formed by all the linear combinations of the vectors of U_1 and U_2, i.e., the linear hull of $U_1 \cup U_2$. Formally, $U_1 + U_2 = \{u_1 + u_2 | u_1 \in U_1, u_2 \in U_2\}$.

2.2 Branch-Decompositions

A *branch-decomposition* of a matroid $\mathcal{M}$ with ground set E is a tree T such that

- all inner nodes of T have degree three, and
- the leaves of T one-to-one correspond to the elements of $\mathcal{M}$.

An edge e of T splits T into two subtrees and the elements corresponding to the leaves of the two subtrees form a partition (E_1, E_2) of the ground set E. The *width of an edge e* is equal to $r(E_1)+r(E_2)-r(E)$, i.e., to the smallest k such that (E_1, E_2) is a $(k+1)$-separation of $\mathcal{M}$. The *width of the branch-decomposition T* is the maximum width of an edge e of T. Finally, the *branch-width* of a matroid is the minimum width of a branch-decomposition of $\mathcal{M}$ and is denoted by $\mathrm{bw}(\mathcal{M})$.

2.3 Decomposition Width

We now formally define our new width parameter. A *K-decomposition*, $K \geq 1$, of a matroid $\mathcal{M}$ with ground set E is a rooted tree $\mathcal{T}$ such that

- the leaves of $\mathcal{T}$ one-to-one correspond to the elements of E,
- some leaves of T are special (we refer to them as "special leaves"), and
- each inner node v of $\mathcal{T}$ has exactly two children and is associated with two functions φ_v and φ_v^r such that $\varphi_v : \{0,\ldots,K\}\times\{0,\ldots,K\} \to \{0,\ldots,K\}$ and $\varphi_v^r : \{0,\ldots,K\} \times \{0,\ldots,K\} \to \mathbb{N}$ such that $\varphi_v(0,0) = 0$ and $\varphi_v^r(0,0) = 0$.

We now define a function $r_T : 2^E \to \mathbb{N}$ based on the K-decomposition T.

The value $r_T(F)$ for $F \subseteq E$ is determined as follows. Each vertex of T is assigned a color (an integer between 0 and K) and a label (a non-negative integer). A non-special leaf of $\mathcal{T}$ corresponding to an element in F is colored and labelled with 1 and other leaves are colored and labelled with 0. If v is an inner node of $\mathcal{T}$ and its two children are labeled with λ_1 and λ_2 and colored with γ_1 and γ_2, the node v is colored with $\varphi_v(\gamma_1,\gamma_2)$ and labeled with the number $\lambda_1 + \lambda_2 - \varphi_v^r(\gamma_1,\gamma_2)$. Note that if $F = \emptyset$, then all the nodes have color 0 and are labelled with 0. The value $r_T(F)$ is defined to be the label of the root of T. Since the values of φ_v^r for the root v of T do not influence r_T, φ_v^r can be assumed to be identically equal to 0 for the the root v.

If $r_T(F) = r(F)$ for every $F \subseteq E$ where r is the rank function of $\mathcal{M}$, then T *represents* the matroid $\mathcal{M}$. The *decomposition width* of a matroid $\mathcal{M}$ is the smallest K such that there is a K-decomposition representing $\mathcal{M}$. An example of a K-decomposition is given in Figure 1.

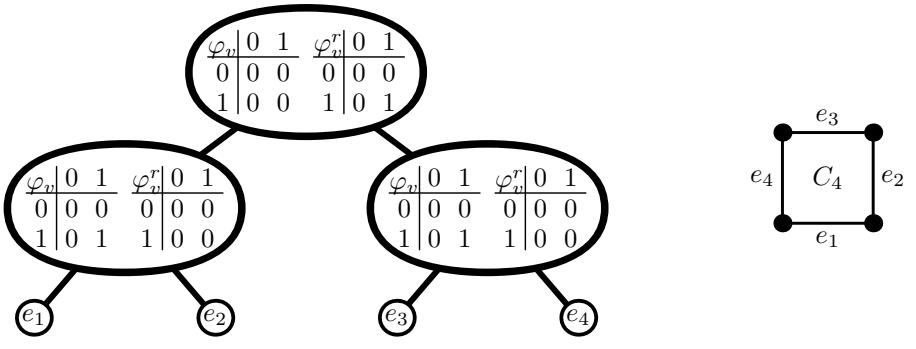


Fig. 1. A 1-decomposition representing a graphic matroid corresponding to the cycle C_4 of length four. None of the leaves is special.

We now give intuitive explanation behind K-decompositions. The colors represent types of different subsets of elements of $\mathcal{M}$, i.e., if E_v is the set of elements assigned to leaves of a subtree rooted at an inner vertex v, then those subsets of E_v that get the same color at v are of the same type. The labels represent the ranks of subsets. Subsets of the same type can have different labels (and thus ranks) but they behave in the same way in the following sense: if E_1 and E_2 are elements assigned to leaves of two subtrees rooted at children of v, then the rank of the union of two subsets of E_1 with the same color and two subsets of

E_2 with the same color is equal to the sum of their ranks decreased by the same amount. Finally, special leaves allows to recognize loop elements of $\mathcal{M}$.

3 Constructing and Verifying Decompositions

We now relate the decomposition width of matroids representable over finite fields to their branch-width. Recall that there is no function of branch-width bounding the decomposition width for general matroids as we have explained in Introduction. The proof is omitted due to space limits.

Theorem 1. *Let $\mathcal{M}$ be a matroid representable over a finite field $\mathbb{F}$ of order q. If the branch-width of $\mathcal{M}$ is at most $k \geq 1$, then the decomposition width of $\mathcal{M}$ is at most $K = \frac{q^{k+1}-q(k+1)+k}{(q-1)^2}$. Moreover, if a branch-decomposition of $\mathcal{M}$ with width k and its representation over $\mathbb{F}$ are given, then a K-decomposition of $\mathcal{M}$ can be constructed in time $O(n^{1+\alpha})$ where n is the number of elements of $\mathcal{M}$ and α is the exponent from the matrix multiplication algorithm.*

Combining Theorem 1 and the cubic-time algorithm of Hliněný and Oum [17] for computing branch-decompositions of matroids with bounded branch-width, we obtain the following:

Corollary 1. *Let $\mathcal{M}$ be a matroid represented over a finite field $\mathbb{F}$ of order q. For every $k \geq 1$, there exists an algorithm running in time $O(n^{1+\alpha})$, where n is the number of elements of $\mathcal{M}$ and α is the exponent from the matrix multiplication algorithm, that either outputs that the branch-width of $\mathcal{M}$ is bigger than k or constructs a K-decomposition representing $\mathcal{M}$ for $K \leq \frac{q^{k+1}-q(k+1)+k}{(q-1)^2}$.*

A K-decomposition fully describes the matroid it represents through functions φ_v and φ_v^r. However, not all possible choices of φ_v and φ_v^r give rise to a decomposition representing a matroid and it is natural to ask whether we can efficiently test that a K-decomposition represents a matroid. We answer this equation in the affirmative way in the next theorem; the proof is omitted due to space constraints.

Theorem 2. *For every K-decomposition $\mathcal{T}$ with n leaves, it can be tested in time $O(K^8 n)$ whether $\mathcal{T}$ corresponds to a matroid.*

4 Computing the Tutte Polynomial

One of the classical polynomials associated to matroids (and graphs) is the Tutte polynomial. There are several equivalent definitions of this polynomial, but we provide here only the one we use. the *Tutte polynomial* $T_{\mathcal{M}}(x, y)$ is equal to

$$T_{\mathcal{M}}(x,y) = \sum_{F \subseteq E} (x-1)^{r(E)-r(F)}(y-1)^{|F|-r(F)} \tag{1}$$

The Tutte polynomial is an important algebraic object associated to a matroid. Some of the values of $T_{\mathcal{M}}(x, y)$ have a combinatorial interpretation; as a simple example, the value $T_{\mathcal{M}}(1, 1)$ is equal to the number of bases of a matroid $\mathcal{M}$. We show that the Tutte polynomial can be computed, i.e., its coefficients can be listed, and evaluated, i.e., its value for given x and y can be determined, in time $O(K^2n^3r^2)$ for n-element matroids of rank r given by the K-decomposition. The proof for computing the Tutte polynomial reflects the main motivation behind the definition of our width parameter.

Theorem 3. *Let K be a fixed integer. The Tutte polynomial of an n-element matroid $\mathcal{M}$ given by its K-decomposition can be computed in time $O(K^2n^3r^2)$ and evaluated in time $O(K^2n)$ where r is the rank of $\mathcal{M}$ (under the assumption that summing and multiplying $O(n)$-bit numbers can be done in constant time).*

Proof. First, we give an algorithm for computing the Tutte polynomial, i.e., an algorithm that computes the coefficients in the polynomial. Let $\mathcal{T}$ be a K-decomposition of $\mathcal{M}$ and let E_v be the elements of $\mathcal{M}$ corresponding to leaves of the subtree rooted at a vertex v. For every node v of $\mathcal{M}$ and every triple $[\gamma, n', r']$, $0 \leq \gamma \leq K$, $0 \leq n' \leq n$ and $0 \leq r' \leq r$, we compute the number of subsets F of E_v such that the color assigned to v for F is γ, $|F| = n'$ and the rank of F is r'. These numbers will be denoted by $\mu_v(\gamma, n', r')$.

The numbers $\mu_v(\gamma, n', r')$ are computed from the leaves towards the root of $\mathcal{T}$. If v is a leaf of $\mathcal{T}$, then $\mu_v(0, 0, 0)$ is equal to 1. The value of $\mu_v(1, 1, 1)$ is also equal to 1 unless v is special; if v is special, then $\mu_v(0, 1, 0) = 1$. All the other values of μ_v are set to 0.

Let v be a node with children v_1 and v_2. If F_i is a subset of E_{v_i} with color γ_i such that $n_i = |F_i|$ and $r_i = r(F_i)$, then $F_1 \cup F_2$ is a subset of E_v with color $\varphi_v(\gamma_1, \gamma_2)$, with $n_1 + n_2$ elements and the rank $r_1 + r_2 - \varphi_v^r(\gamma_1, \gamma_2)$. In other words, it holds that

$$\mu_v(\gamma, n', r') = \sum_{\gamma_1, n_1, r_1, \gamma_2, n_2, r_2} \mu_{v_1}(\gamma_1, n_1, r_1)\mu_{v_2}(\gamma_2, n_2, r_2) \tag{2}$$

where the sum is taken over six-tuples $(\gamma_1, n_1, r_1, \gamma_2, n_2, r_2)$ such that $\gamma = \varphi_v(\gamma_1, \gamma_2)$, $n' = n_1 + n_2$ and $r' = r_1 + r_2 - \varphi_v^r(\gamma_1, \gamma_2)$. Computing μ_v from the values of μ_{v_1} and μ_{v_2} base on (2) requires time $O(K^2n^2r^2)$ and thus the total running time of the algorithm is $O(K^2n^3r^2)$. The Tutte polynomial of $\mathcal{M}$ can be read from μ_r where r is the root of $\mathcal{T}$ since the value $\mu_r(0, \alpha, \beta)$ is the coefficient at $(x-1)^{r(E)-\beta}(y-1)^{\alpha-\beta}$ in (1).

Let us turn our attention to evaluating the Tutte polynomial for given values of x and y. This time, we recursively compute the following quantity for every node v of $\mathcal{T}$:

$$\mu_v(\gamma) = \sum_{F \subseteq E_v} (x-1)^{r(E)-r(F)}(y-1)^{|F|-r(F)} \tag{3}$$

where the sum is taken over the subsets F with color γ. The value of $\mu_v(0)$ is equal to $(x-1)^{r(E)}y$ for special leaves v and to $(x-1)^{r(E)}$ for non-special leaves

v. For non-special leaves, $\mu_v(1)$ is equal to $(x-1)^{r(E)-1}$. All the other values of μ_v are equal to 0.

For a node v of $\mathcal{T}$ with two children v_1 and v_2, the equation (3) and the definition of a K-decomposition implies that

$$\mu_v(\gamma) = \sum_{0 \leq \gamma_1, \gamma_2 \leq K} \frac{\mu_{v_1}(\gamma_1)\mu_{v_2}(\gamma_2)}{(x-1)^{r(E)-\varphi_v^r(\gamma_1,\gamma_2)}(y-1)^{-\varphi_v^r(\gamma_1,\gamma_2)}}$$

where the sum is taken over values γ_1 and γ_2 such that $\gamma = \varphi_v(\gamma_1, \gamma_2)$. Under the assumption that arithmetic operations require constant time, determining the values of μ_v needs time $O(K^2)$. Since the number of nodes of $\mathcal{T}$ is $O(n)$, the total running time of the algorithm is $O(K^2 n)$ as claimed in the statement of the theorem.

As a corollary, we obtain Hliněný's result on computing the Tutte polynomial and its values for matroids represented over finite fields of bounded branch-width [16].

Corollary 2. *Let $\mathbb{F}$ be a fixed finite field and k a fixed integer. There is a polynomial-time algorithm for computing and evaluating the Tutte polynomial for the class of matroids of branch-width at most k representable over $\mathbb{F}$ that are given by their representation over the field $\mathbb{F}$.*

5 Deciding MSOL-Properties

In this section, we show that there is a linear-time algorithm for deciding monadic second order logic formulas for matroids of bounded decomposition width when the decomposition is given as part of input.

First, let us be precise on the type of formulas that we are interested in. A *monadic second order logic formula* ψ, an MSOL formula, for a matroid contains basic logic operators (the negation, the disjunction and the conjunction), quantifications over elements and sets of elements of a matroid (we refer to these variables as to element and set variables), the equality predicate, the predicate of containment of an element in a set and the independence predicate which determines whether a set of elements of a matroid is independent. The independence predicate depends on and encodes an input matroid $\mathcal{M}$. Properties expressible in the monadic second order logic include many well-known NP-complete problems, e.g., it can be expressed whether a graphic matroid corresponds to a hamiltonian graph.

In order to present the algorithm, we introduce auxiliary notions of a K-half and K-halved matroids which we extend to interpreted K-halves. A *K-half* is a matroid $\mathcal{M}$ with ground-set E and equipped with a function $\varphi_{\mathcal{M}} : 2^E \to \{0, \ldots, K\}$. A *$K$-halved* matroid is a matroid $\mathcal{M}$ with ground-set $E = E_1 \cup E_2$ composed of a K-half $\mathcal{M}_1$ with ground set E_1 and a matroid $\mathcal{M}_2$ with ground set E_2 such that each subset of $F \subseteq E_2$ is assigned a vector w^F of $K+1$ non-negative integers. Both $\mathcal{M}_1$ and $\mathcal{M}_2$ are matroids. The rank of a subset $F \subseteq E_1 \cup E_2$ is given by the following formula:

$$r_{\mathcal{M}}(F) = r_{\mathcal{M}_1}(F \cap E_1) + w^{F \cap E_2}_{\varphi_{\mathcal{M}_1}(F \cap E_1)} ,$$

where $w^{F\cap E_2}_{\varphi_{\mathcal{M}_1}(F\cap E_1)}$ is the coordinate of the vector $w^{F\cap E_2}$ corresponding to the value of $\varphi_{\mathcal{M}_1}(F\cap E_1)$. We will write $\mathcal{M} = \mathcal{M}_1 \oplus_K \mathcal{M}_2$ to represent the fact that the matroid $\mathcal{M}$ is a K-halved matroid obtained from $\mathcal{M}_1$ and $\mathcal{M}_2$ in the way we have just described.

The next lemma justifies the definition of K-halved matroids: it asserts that every matroid $\mathcal{M}$ represented by a K-decomposition can be viewed as a composed of two K-halves, one of which is $\mathcal{M}$ restricted to the elements corresponding to leaves of a subtree of its K-decomposition. The proof is omitted due to space constraints.

Lemma 1. *Let $\mathcal{T}$ be a K-decomposition of a matroid $\mathcal{M}$, $K \geq 1$, and let v be a node of $\mathcal{T}$. Further, let E_v be the set of elements of $\mathcal{M}$ assigned to the leaves of the subtree of $\mathcal{T}$ rooted at v, and $\overline{E_v}$ the set of the remaining elements of $\mathcal{M}$. If F_1 and F_2 are two subsets of E_v such that the color of v with respect to the decomposition is the same for F_1 and F_2, then*

$$r(F) + r(F_1) - r(F \cup F_1) = r(F) + r(F_2) - r(F \cup F_2), \text{ i.e.,}$$

$$r(F \cup F_1) - r(F_1) = r(F \cup F_2) - r(F_2),$$

for every subset F of $\overline{E_v}$.

For an MSOL formula ψ with no free variables, two K-halves $\mathcal{M}_1$ and $\mathcal{M}'_1$ are *ψ-equivalent* if the formula ψ is satisfied for $\mathcal{M}_1 \oplus_K \mathcal{M}_2$ if and only if ψ is satisfied for $\mathcal{M}'_1 \oplus_K \mathcal{M}_2$. It can be shown that the number of ψ-equivalence classes of K-halves is finite.

Lemma 2. *Let ψ be a fixed MSOL formula and K a fixed integer. The number of ψ-equivalence classes of K-halves is finite.*

For a K-decomposition $\mathcal{T}$ of a matroid $\mathcal{M}$, we can obtain a K-half by restricting $\mathcal{M}$ to the elements corresponding to the leaves of a subtree of $\mathcal{T}$ (note that the subsets of the elements not corresponding to the leaves of $\mathcal{T}$ can be assigned non-negative integers as in the definition of a K-halved matroid by Lemma 1). The K-half obtained from $\mathcal{M}$ by restricting it to the elements corresponding to the leaves of a subtree rooted at a vertex v of $\mathcal{T}$ is further denoted by $\mathcal{M}_v$. The next lemma is the core of the linear-time algorithm for deciding the satisfiability of the formula ψ.

Lemma 3. *Let ψ be a monadic second order logic formula, let $\mathcal{T}$ be a K-decomposition of a matroid $\mathcal{M}$ and let v be a node of $\mathcal{T}$ with children v_1 and v_2. The ψ-equivalence class of $\mathcal{M}_v$ is uniquely determined by the ψ-equivalence classes of $\mathcal{M}_{v_1}$ and $\mathcal{M}_{v_2}$, and the functions φ_v and φ^r_v.*

We are now ready to present the main result of this section.

Theorem 4. *Let ψ be a fixed monadic second order logic and K a fixed integer. There exists an $O(n)$-time algorithm that given an n-element matroid $\mathcal{M}$ with its K-decomposition decides whether $\mathcal{M}$ satisfies ψ.*

Proof. The core of our algorithm is Lemma 3. Since ψ and K are fixed and the number of ψ-equivalence classes of K-halves is finite by Lemma 2, we can wire in the algorithm the transition table from the equivalence classes of $\mathcal{M}_{v_1}$ and $\mathcal{M}_{v_2}$, φ_v and φ_v^r to the equivalence class of $\mathcal{M}_v$ where v is a node of $\mathcal{T}$ and v_1 and v_2 its two children. At the beginning, we determine the equivalence classes of $\mathcal{M}_v$ for the leaves v of $\mathcal{T}$; this is easy since the equivalence class of $\mathcal{M}_v$ for a leaf v depends only on the fact whether the element corresponding to v is a loop or not.

Then, using the wired in transition table, we determine in constant time the equivalence class of $\mathcal{M}_v$ for each node v based on φ_v, φ_v^r and the equivalence classes of $\mathcal{M}_{v_1}$ and $\mathcal{M}_{v_2}$ for children v_1 and v_2 of v. Observe that the definition of ψ-equivalence implies that the equivalence classes of $\mathcal{M}_{v_1}$ and $\mathcal{M}_{v_2}$ where v_1 and v_2 are the two children of the root determine whether ψ is satisfied for $\mathcal{M}$. Hence, once the equivalence classes of $\mathcal{M}_{v_1}$ and $\mathcal{M}_{v_2}$ are found, it is easy to determine whether $\mathcal{M}$ satisfies ψ using another wired-in table.

As K and ψ are fixed, the running time of our algorithm is clearly linear in the number of nodes of $\mathcal{T}$ which is linear in the number of elements of the matroid $\mathcal{M}$.

Corollary 1 and Theorem 4 yield the following result originally proved by Hliněný [14]:

Corollary 3. *Let $\mathbb{F}$ be a fixed finite field, k a fixed integer and ψ a fixed monadic second order logic formula. There is a polynomial-time algorithm for deciding whether a matroid of branch-width at most k given by its representation over the field $\mathbb{F}$ satisfies ψ.*

6 Deciding Representability

We adopt the algorithm presented in [20]. Let us start with recalling some definitions from [20]. A *rooted branch-decomposition* of $\mathcal{M}$ is obtained from a branch-decomposition of $\mathcal{M}$ by subdividing an edge and rooting the decomposition tree at a new vertex. If $\mathcal{M}$ is a matroid and (A, B) is a partition of its ground set, then two subsets A_1 and A_2 are *B-indistinguishable* if the following identity holds for every subset B' of B:

$$r(A_1 \cup B') - r(A_1) = r(A_2 \cup B') - r(A_2) \; .$$

Clearly, the relation of being B-indistinguishable is an equivalence relation on subsets of A. Finally, the *transition matrix* for an inner node v of a rooted branch-decomposition $\mathcal{M}$ is the matrix whose rows correspond to $\overline{E_1}$-indistinguishable subsets of E_1 and columns to $\overline{E_2}$-indistinguishable subsets of E_2 where E_1 and E_2 are the elements corresponding to the leaves of the two subtrees rooted at the children of v and $\overline{E_1}$ and $\overline{E_2}$ are their complements. The entry of $\mathcal{M}$ in the row corresponding to the equivalence class of $F_1 \subseteq E_1$ and in the column corresponding to the equivalence class of $F_2 \subseteq E_2$ is equal to $r(F_1) + r(F_2) - r(F_1 \cup F_2)$. By the definition of indistinguishability, the value of the entry is independent of the choice of F_1 and F_2 in their equivalence classes.

The main algorithmic result of [20] can be reformulated as follows:

Theorem 5. *Let k be a fixed integer and q a fixed prime power. There is a polynomial-time algorithm that for a matroid $\mathcal{M}$ given by its rooted branch-decomposition with transition matrices whose number of rows and columns is at most k and a (oracle-given) mapping of subsets to equivalence classes corresponding to rows and columns of its matrices decides whether $\mathcal{M}$ can be represented over $\mathbb{GF}(q)$ and if so, it computes one of its representations over $\mathbb{GF}(q)$. Moreover, the algorithm can be modified to count all non-isomorphic representations of $\mathcal{M}$ over $\mathbb{GF}(q)$ and to list them (in time linearly dependent on the number of non-isomorphic representations).*

Let $\mathcal{T}$ be a K-decomposition of a matroid $\mathcal{M}$, v an inner node of $\mathcal{T}$ and E_v the subset of the ground set of $\mathcal{M}$ containing the elements corresponding to the leaves of subtree of $\mathcal{T}$ rooted at v. View $\mathcal{T}$ as a rooted branch-decomposition of $\mathcal{M}$. Observe that any two subsets of E_v assigned the same color at v are $\overline{E_v}$-indistinguishable where $\overline{E_v}$ is the complement of E_v. In addition, the values of the function φ_v^r are entries of the transition matrix at v as defined in the beginning of this section. Hence, Theorem 5 yields the following.

Corollary 4. *For every integer K and every prime power q, there is a polynomial-time algorithm that for a matroid $\mathcal{M}$ given by its K-decomposition, decides whether $\mathcal{M}$ can be represented over $\mathbb{GF}(q)$ and if so, it computes one of its representations over $\mathbb{GF}(q)$. Moreover, the algorithm can be modified to count all non-isomorphic representations of $\mathcal{M}$ over $\mathbb{GF}(q)$ and to list them (in time linearly dependent on the number of non-isomorphic representations).*

Acknowledgement

The author would like to thank Bruno Courcelle for his valuable comments on the conference paper [20] and Ondřej Pangrác for his insightful comments on the content of this paper as well as careful reading its early version. The comments of the three anonymous referees helped to improve and clarify the presentation a lot and are also greatly appreciated.

References

1. Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of fiding embeddings in a k-tree. SIAM J. Alg. Disc. Meth. 8, 277–284 (1987)
2. Arnborg, S., Lagergren, J., Seese, D.: Easy problems for tree decomposable graphs. J. Algorithms 12, 308–340 (1991)
3. Bodlaender, H.: Dynamic programming algorithms on graphs with bounded tree-width. In: Lepistö, T., Salomaa, A. (eds.) ICALP 1988. LNCS, vol. 317, pp. 105–119. Springer, Heidelberg (1988)
4. Bodlaender, H.: A linear time algorithm for finding tree-decompositions of small treewidth. In: Proc. SODA 1993, pp. 226–234. ACM & SIAM (1993)
5. Courcelle, B.: The monadic second-order logic of graph I. Recognizable sets of finite graphs. Inform. and Comput. 85, 12–75 (1990)

6. Courcelle, B.: The expression of graph properties and graph transformations in monadic second-order logic. In: Rozenberg, G. (ed.) Handbook of graph grammars and computing by graph transformations. Foundations, vol. 1, pp. 313–400. World Scientific, Singapore (1997)
7. Geelen, J., Gerards, B., Whittle, G.: Branch-width and well-quasi-ordering in matroids and graphs. J. Combin. Theory Ser. B 84, 270–290 (2002)
8. Geelen, J., Gerards, B., Whittle, G.: On Rota's Conjecture and excluded minors containing large projective geometries. J. Combin. Theory Ser. B 96, 405–425 (2006)
9. Geelen, J., Gerards, B., Whittle, G.: Excluding a planar graph from GF(q)-representable matroids. J. Combin. Theory Ser. B 97, 971–998 (2007)
10. Geelen, J., Gerards, B., Whittle, G.: Tangles, tree-decompositions, and grids in matroids. J. Combin. Theory Ser. B 97, 657–667 (2009)
11. Hliněný, P.: Branch-width, parse trees, and monadic second-order logic for matroids. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 319–330. Springer, Heidelberg (2003)
12. Hliněný, P.: On matroid properties definable in the MSO logic. In: Rovan, B., Vojtáš, P. (eds.) MFCS 2003. LNCS, vol. 2747, pp. 470–479. Springer, Heidelberg (2003)
13. Hliněný, P.: A parametrized algorithm for matroid branch-width. SIAM J. Computing 35, 259–277 (2005)
14. Hliněný, P.: Branch-width, parse trees, and monadic second-order logic for matroids, J. Combin. Theory Ser. B 96, 325–351 (2006)
15. Hliněný, P.: On matroid representability and minor problems. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 505–516. Springer, Heidelberg (2006)
16. Hliněný, P.: The Tutte polynomial for matroids of bounded branch-width. Combin. Probab. Comput. 15, 397–406 (2006)
17. Hliněný, P., Oum, S.: Finding branch-decomposition and rank-decomposition. SIAM J. Computing 38, 1012–1032 (2008)
18. Hliněný, P., Whittle, G.: Matroid tree-width. Europ. J. Combin. 27, 1117–1128 (2006)
19. Hliněný, P., Whittle, G.: Addendum to Matroid tree-Width. Europ. J. Combin. 30, 1036–1044 (2009)
20. Král', D.: Computing representations of matroids of bounded branch-width. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 224–235. Springer, Heidelberg (2007)
21. Oum, S., Seymour, P.D.: Approximating clique-width and branch-width. J. Combin. Theory Ser. B 96, 514–528 (2006)
22. Oum, S., Seymour, P.D.: Testing branch-width. J. Combin. Theory Ser. B 97, 385–393 (2007)
23. Oxley, J.G.: Matroid theory. Oxford Graduate Texts in Mathematics, vol. 3. Oxford University Press, Oxford (1992)
24. Seymour, P.: Recognizing graphic matroids. Combinatorica 1, 75–78 (1981)
25. Strozecki, Y.: A logical approach to decomposable matroids, arXiv 0908.4499
26. Truemper, K.: Matroid decomposition. Academic Press, London (1992)

The Cooperative Game Theory Foundations of Network Bargaining Games*

MohammadHossein Bateni**, MohammadTaghi Hajiaghayi,
Nicole Immorlica, and Hamid Mahini***

Princeton University, Princeton, NJ
mbateni@cs.princeton.edu
AT&T Labs–Research, Florham Park, NJ
hajiagha@research.att.com
Northwestern University, Chicago, IL
nickle@eecs.northwestern.edu
Sharif University of Technology, Tehran, Iran
mahini@ce.sharif.edu

Abstract. We study bargaining games between suppliers and manufacturers in a network context. Agents wish to enter into contracts in order to generate surplus which then must be divided among the participants. Potential contracts and their surplus are represented by weighted edges in our bipartite network. Each agent in the market is additionally limited by a capacity representing the number of contracts which he or she may undertake. When all agents are limited to just one contract each, prior research applied natural generalizations of the Nash bargaining solution to the networked setting, defined the new solution concepts of *stable* and *balanced*, and characterized the resulting bargaining outcomes. We simplify and generalize these results to a setting in which participants in only one side of the market are limited to one contract each. The core of our results uses a linear-programming formulation to establish a novel connection between well-studied cooperative game theory concepts and the solution concepts of *core* and *prekernel* defined for the bargaining games. This immediately implies one can take advantage of the results and algorithms in cooperative game theory to reproduce results such as those of Azar et al. [1] and Kleinberg and Tardos [28] and generalize them to our setting. The cooperative-game-theoretic connection also inspires us to refine our solution space using standard solution concepts from that literature such as *nucleolus* and *lexicographic kernel*. The nucleolus is particularly attractive as it is unique, always exists, and is supported by experimental data in the network bargaining literature. Guided by algorithms from cooperative game theory, we show how to compute the nucleolus by pruning and iteratively solving a natural linear-programming formulation.

* The full version of this extended abstract is available as [3], which contains all the missing proofs as well as more discussion about the results.

** The author is also with Center for Computational Intractability, Princeton, NJ 08540. He was supported by a Gordon Wu fellowship as well as NSF ITR grants CCF-0205594, CCF-0426582 and NSF CCF 0832797, NSF CAREER award CCF-0237113, MSPA-MCS award 0528414, NSF expeditions award 0832797.

*** Most of the work was done while the author was visiting Northwestern University, Chicago, IL. He was supported in part by Iranian National Elites' Foundation.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 67–78, 2010.

1 Introduction

Common wisdom has it that the whole is more than the sum of the parts. In more economic terms, this proverb is a translation of the fact that two cooperative agents are often capable of generating a surplus that neither could achieve alone. For example, the owner of a music studio, together with a music band, can record and sell an album; a publishing house, together with an author, can print and sell a book. The proceeds from the album or the book are the surplus generated by the cooperation of the agents. *Bargaining theory* asks how agents divide this jointly generated surplus.

In the 1950s, John Nash proposed a solution to this problem for the special case of two agents [32]. His solution, known as the *Nash bargaining solution*, is based on the intuition that, all else being equal, agents will tend to divide the surplus equally. If things are not equal, i.e., if some agents have better outside options than others, then the *net* surplus will be divided equally. Since Nash's result, various economists and computer scientists have extended this solution concept to a networked setting and defined new equilibrium concepts as well [6,28]. In these settings, each agent has a set of potential contracts, represented by the network. The "outside options" of the Nash bargaining solution are now endogenous to the model and generated by the network structure itself.

We propose looking at the network bargaining game through the lens of *cooperative game theory*. In a cooperative game, sets of agents are mapped to values representing the surplus they alone can generate. A solution then assigns a payoff to each agent. The literature has a rich history exploring various solution concepts, their axiomatic properties, and their computability. By interpreting the network bargaining game in this context, we are able to leverage the valuable tools and concepts of cooperative game theory when studying network bargaining. Not only does this enable us to reproduce previous results (with arguably little effort) and derive these previous results for more general models, but more importantly perhaps, we are introduced to new refined solution concepts such as *nucleolus* and *lexicographic kernel*. These concepts are often arguably more predictive than those previously studied.

Our Results. In most prior work on networked bargaining, researchers assume each agent can enter *at most one* contract, i.e., the set of contracts form a *matching*. Additionally, contracts are often assumed to be of equal worth. In this paper, we generalize these models by assigning each agent a capacity constraint and allowing him or her to participate in a number of contracts limited by this capacity. We also allow contracts to generate varying amounts of surplus. We mainly focus our efforts on the important special case of bipartite networks, or networks like the music and literature examples above, in which each agent can be divided into one of two types depending on which type of agent he or she contracts with.

The most basic question in these models is to develop predictions regarding which contracts will form and how much each agent will earn as a result. A successful prediction should be *intuitive*, *computationally tractable*, and *unique*. To this end, Kleinberg and Tardos [28] proposed a solution concept for the matching case called *stable* and a refinement called *balanced* that are natural generalizations of the Nash bargaining solution for the two-player case [32].

We first show how to characterize all stable solutions in our general setting using a linear-programming formulation which is a generalization of one developed by Shapley and Shubik [36] for the matching case. We then introduce a special case of our problem in which one side of the market is severely capacity-constrained. In particular, we assume agents on one side of the market can enter into *at most one* contract each while the other side has general constraints. In this constrained setting, we draw connections between balanced and stable outcomes and the cooperative game-theoretic notions of *core* and *prekernel*. These notions look for solutions that are stable with respect to deviating coalitions of any size. Unlike the general setting, in this setting we prove that the set of stable solutions and the core coincide, as does the set of balanced solutions and the prekernel. This result is of particular interest as the core and prekernel are axiomatic solution concepts of exponential description size (essentially they require that certain conditions hold for *every* subset of agents), whereas the notions of stable and balanced solutions have inherently polynomial descriptions. These connections allow us to leverage existing results in the cooperative game theory literature (such as those for computation of core, prekernel, nucleolus, and lexicographic kernel) to motivate and derive solutions in our setting.

As for leveraging tools from cooperative game theory, the connections we draw imply that the techniques of Faigle, Kern, and Kuipers [19] for finding a point in the prekernel of a game can be adapted to find a balanced solution for our constrained bargaining game. Indeed, the resulting algorithm as well as its analysis is essentially the same as the local dynamics given in Azar et al. [1]. These connections also have implications for the model of Kleinberg and Tardos [28]. In their model, the set of possible contracts is not necessarily bipartite, but instead each agent is restricted to participate in at most one contract. Our aforementioned results regarding stable and balanced solutions can be adapted to this setting. Since the set of stable solutions and core coincide, we are able to characterize all graphs which have at least one stable outcome. Namely, a graph has a stable outcome if and only if the simple maximum-weight matching LP relaxation has integrality gap one. Since the set of balanced solutions and the prekernel coincide, we can obtain the Kleinberg-Tardos result for constructing a balanced outcome in polynomial time using simple and well-known results from the economics literature rather than combinatorial constructs like posets and Edmonds-Gallai decompositions.

Perhaps more importantly, this connection to cooperative game theory guides us in our search for solution concepts for our game. The set of stable/balanced and core/kernel solutions previously proposed may be quite large and hence not terribly predictive. With the goal of computing a unique and predictive outcome for our game, we propose and motivate the cooperative game-theoretic notion of *nucleolus* as the "right" outcome. The nucleolus is unique and symmetric in that agents with similar opportunities have similar earnings. It is also supported by economic experiments, as discussed in Section 2.2. Additionally, for the above model, we show that the nucleolus is computationally tractable. We prove this by following an iterative linear-programming based approach used previously by economists [23,17,40,19] and computer scientists [15] in unrelated contexts. In order to adopt this approach to our setting, we show how to prune the linear programs, creating programs of polynomial size. The main technical difficulty is to prove that this

pruning maintains the essential relationships between the iterated linear programs and thus still computes the nucleolus.

Related work. The most closely related work to ours is that of Kleinberg and Tardos [28]. That paper defines stable and balanced solutions for the matching case mentioned above. They then give an efficient characterization of stable/balanced solutions based on posets and the Edmonds-Gallai decomposition. Our work re-derives and generalizes some of their results using simple and well-known results from the economics literature. Very recently, Azar et al. [1] show that local dynamics does in fact converge to a stable and balanced solution. Incidentally, the connection that we establish between the solution concepts of prekernel and balanced would immediately imply the same local algorithm via a former result of Stearns [41]. We also learned of two other very recent results: Kanoria et al. [26] addresses the problem of finding a "natural" local dynamics for this game, and Azar, Devanur, Jain, and Rabani [2] also study a special case of our problem through cooperative game theory and propose nucleolus as a plausible outcome of the networked bargaining game. The work of Chakraborty et al. [7] as well as that of Chakraborty and Kearns [6] considers a related problem, in which there is no capacity constraints on the vertices but agents have non-linear utilities. They explore the existence and computability of the Nash bargaining solution as well as the proportional bargaining solution in their setting.

Much recent literature has focused on the computability of various solution concepts in the economics literature. In the non-cooperative game theoretic setting, the complexity of Nash and approximate Nash equilibria has a rich recent history [42, 29, 12, 9, 10, 5, 13]. In cooperative game-theoretic settings, the core of a game defined by a combinatorial optimization problem is fundamentally related to the integrality gap of a natural linear program, as observed in numerous prior work [4, 37, 25, 30, 24]. The computability of the nucleolus has also been studied for some special games [15, 27, 40, 19, 20, 18, 23].

2 Preliminaries

In the network bargaining game, there is a set N of n agents. For bipartite graphs, the set N is partitioned into two disjoint sets V_1 and V_2 (i.e., $N = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$) and all edges of the network pair one vertex of V_1 with one vertex of V_2. Each agent $i \in N$ is assigned a *capacity* c_i limiting the number of contracts in which agent i may participate. For each pair of agents $i, j \in N$, we are given a weight w_{ij} representing the *surplus* of a contract between i and j (a weight of $w_{ij} = 0$ means i and j are unable to contract with each other). The capacities together with the weights jointly define a node-and-edge-weighted graph $G = (N, E)$ where $E = \{(i, j) : w_{ij} > 0\}$, the weight of edge (i, j) is w_{ij}, and the weight of node i is c_i. Our game is fully defined by this construct.

The *(bipartite) bargaining game* is a (bipartite) graph $G = (N, E)$ together with a set of node capacities $\{c_i\}$ and edge weights $\{w_{ij}\}$. There are two special cases of the above game that we consider separately. The first is the *matching game* in which $c_i = 1$ for all $i \in N$ (note the graph need not be bipartite in the matching game). The matching game was studied by Kleinberg and Tardos [28] in the context of bargaining, as well

as many economists in the context of cooperative game theory [27,17,40]. The second special case is the *constrained bipartite game* in which the graph $G = (V_1 \cup V_2, E)$ is bipartite and the capacities of all agents on one side of the market are one ($c_i = 1$ for all $i \in V_2$).

2.1 Solution Concepts

Our main task is to predict the set of contracts $M \subseteq E$ and the division of surplus $\{z_{ij}\}$ that result from bargaining among agents. We call a set of contracts M *feasible* if each node i is in at most c_i contracts: i.e., for each $i \in N$, $|\{j : (i,j) \in M\}| \leq c_i$. A *solution* $(\{z_{ij}\}, M)$ of a bargaining game is a division of surplus $\{z_{ij}\}$ together with a set of feasible contracts M such that the total surplus generated is divided among the agents involved: i.e., for all $(i,j) \in M$, $z_{ij} + z_{ji} = w_{ij}$, and for all $(i,j) \notin M$, $z_{ij} = 0$. We interpret z_{ij} as the amount of money i earns from contracting with j. We also define the aggregate earnings of node i by $x_i = \sum_{j \in N} z_{ij}$ and sometimes refer to the set of earnings $\{x_i\}$ as the *outcome* of our game.

The set of solutions of our game is quite large, and so it is desirable to define a subset of solutions that are likely to arise as a result of the bargaining process. There are two approaches one might take in this endeavor. The first is to generalize the bargaining notions introduced by Nash [32] and later extended to networked settings [28]. The second is to study our game from the perspective of cooperative game theory.

In keeping with the bargaining line of work, we define the *outside option* of an agent i to be the best deal he or she could make with someone outside the contracting set M. For a fixed agent k with $(i,k) \in E \setminus M$, the best deal i can make with k is to match k's current worst offer. If k is under-capacitated in M, i can offer k essentially 0 and so i's outside option with k would be w_{ik}. If k is utilized to capacity, then so long as i offers k at least the minimum of z_{kj} over all j such that $(j,k) \in M$, then k will accept the offer. Generalizing this, we get the following definition.

Definition 1. *The* outside option α_i *of agent* i *in solution* $(\{z_{ij}\}, M)$ *is* $\max_{k:(i,k)\in E\setminus M} \max_{j:(j,k)\in M}(w_{ik} - I_k z_{kj})$, *where* I_k *is a zero-one indicator variable for whether* k *is utilized to capacity. If the set* $\{k : (i,k) \in E \setminus M\}$ *is empty, we define the outside option of* i *to be zero. The inner maximization is defined to be* w_{ik} *if its support set is empty.*

Intuitively, if an agent has a deal in which he or she earns less than the outside option, then he or she will switch to the other contract. Hence, we call a solution *stable* if each agent earns at least the outside option.

Definition 2. *A solution is* stable *if for all* $(i,k) \in M$, $z_{ik} \geq \alpha_i$, *and* $\alpha_i = 0$ *if* i *has residual capacity.*

Nash [32] additionally argued that agents tend to split surplus equally. If agents are heterogeneous in that they have different outside options, then they will split the net surplus equally. Using the terminology of Kleinberg and Tardos [28], we define a solution to be *balanced* if it satisfies Nash's conditions where outside options are defined according to the network structure.

Definition 3. *A solution is* balanced *if for all* $(i,k) \in M$, $z_{ik} - \alpha_i = z_{ki} - \alpha_k$ *or equivalently* $z_{ik} = \alpha_i + \frac{w_{ik}-(\alpha_i+\alpha_k)}{2}$.

Another approach to refining the set of solutions for our game is to study it from the cooperative game theory perspective. A cooperative game is defined by a set of agents N and a value function $\nu : 2^N \rightarrow \Re^+ \cup \{0\}$ mapping subsets of agents to the non-negative real numbers. Intuitively, the value of a set of agents represents the maximum surplus they alone can achieve. Cooperative game theory suggests that the total earnings of agents in a cooperative game is fundamentally related to the values of the sets in which they are contained. To cast our game in the cooperative game theory terminology, we must first define the value of a subset of agents. We will define this to be the best set of contracts they alone can achieve.

Definition 4. *The value* $\nu(S)$ *of a subset* $S \subseteq N$ *of agents is the maximum* $\sum_{(i,j)\in M} w_{ij}$ *over all feasible sets of contracts* M *such that* $i,j \in S$ *for all* $(i,j) \in M$.

In graph-theoretic terminology, this is simply the maximum weighted f-factor[1] of the subgraph restricted to S and can be computed in polynomial time.

Cooperative game theory suggests that each set of agents should earn in total at least as much as they alone can achieve. In mathematical terms, we require that the sum of the earnings of a set of agents should be at least the value of that same set. We additionally require that the total surplus of all agents is exactly divided among the agents. These requirements together yield the cooperative game-theoretic notion of the *core* [22,33].

Definition 5. *An outcome* $\{x_i\}$ *is in the* core *if for all subsets* $S \subseteq N$, $\sum_{i\in S} x_i \geq \nu(S)$, *and for the grand coalition* N, $\sum_{i\in N} x_i = \nu(N)$.

The core may be empty even for very simple classes of games, and it may be hard to test whether it is empty or not [11]. However, for our games, we are able to characterize the set of matching games having a non-empty core and show that all bipartite bargaining games have a non-empty core.

Other solution concepts proposed in the cooperative game theory literature are that of *kernel* and *prekernel* [14]. Unlike the core, the kernel and prekernel always exist. As these concepts are closely related and we only work with the prekernel, we only define the prekernel in this paper.[2] The prekernel is defined by characterizing the power of agent i over agent j, and requiring that these powers are in some sense equalized. Intuitively, the *power* of i over j is the maximum amount i can earn without the cooperation of j.

[1] Given a graph $G(V,E)$ and a function $f : V \mapsto \mathbb{Z}^{\geq 0}$, an *$f$-factor* is a subset $F \subseteq E$ of edges such that each vertex v has exactly $f(v)$ edges incident on it in F. See West [43] for a discussion and for a polynomial-time algorithm to find an f-factor. The approach can be extended to the case where $f(v)$ values are upper bounds on the degrees, and we are interested in finding the maximum-weight solution.

[2] In fact, kernel and prekernel coincide in our game because $\nu(\{i\}) = 0$ for any $i \in N$—indeed, the two closely-related solution concepts coincide for any *zero-monotonic TU-game* [31], and our game is one of this class.

Definition 6. *The* power *of agent i with respect to agent j in the outcome $\{x_i\}$ is*

$$s_{ij}(x) = \max \left\{ \nu(S) - \sum_{k \in S} x_k : S \subseteq N, S \ni i, S \not\ni j \right\}.$$

The prekernel *is the set of outcomes x that satisfy $s_{ij}(x) = s_{ji}(x)$ for every i and j.*

Although the definition of the prekernel is not completely intuitive, it turns out to be similar to the notion of balanced solutions in certain networked settings. A further refinement of this definition is that of *lexicographic kernel* which is, roughly speaking, a subset of the prekernel that lexicographically maximizes the vector of all s_{ij} values. In some sense, this definition tries to be as impartial as possible to different players. As the *nucleolus* defined below is a more widely accepted solution concept and achieves complete impartiality, we do not give detailed information about the lexicographic kernel. We simply note that it has been studied in [21,44], and the result of [21] allows us to compute the lexicographic kernel for any general bipartite (or even non-bipartite) bargaining game.

2.2 A Unique Outcome

None of the solution concepts proposed above are unique. For any given game instance, and any of the solution concepts above, there may be many outcomes which satisfy it. For example, consider a bipartite bargaining game with two vertices on each side. Each of the four possible edges has value one, and the capacities of the vertices are also one. It can be easily verified that any solution assigning value x to the vertices of one side and $1 - x$ to the other side for $0 \leq x \leq 1$, is a stable, balanced solution, and hence (as we will show later) also in the core and kernel. However, the solution corresponding to $x = 0$ seems, in some sense unfair. After all, all agents appear symmetric, as can be formalized by the fact that there exists an automorphism of the game mapping any agent into any other agent. Hence, we expect the agents to have similar earnings after the bargaining procedure. Among all the plausible solution concepts, the one we expect to see is that for which $x = 1/2$, i.e., each agent earns $1/2$. This intuition was tested and verified in the laboratory experiments of Charness et al. [8]. The solution $x = 1/2$ turns out to be the *nucleolus* of the example game. In general, the nucleolus is that outcome which maximizes, lexicographically, the excess earnings of any given set.

Definition 7. *Given an outcome $\{x_i\}$, the* excess $\epsilon(S)$ *of a set S is the extra earnings of S in $\{x_i\}$: $\epsilon(S) = \sum_{i \in S} x_i - \nu(S)$. Let $\epsilon = (\epsilon_{S_1}, \ldots, \epsilon_{S_{2^N}})$ be the vector of excesses sorted in non-decreasing order. The* nucleolus *of a bargaining game is the outcome which maximizes, lexicographically, this vector ϵ.*

The nucleolus was first introduced by Schmeidler [35]. It is a point in the kernel [35], and also part of the core if it is non-empty. We will show later that any point in the core intersect kernel must be stable and balanced (at least for the matching and constrained games), and hence the nucleolus inherits all the nice properties of stable and balanced solutions. In addition, the nucleolus is unique [16], and is in fact characterized by a set of simple, reasonable axioms [34,38,39] including our intuitive notion of symmetry mentioned above.

2.3 Results

In this paper, we are primarily interested in developing natural solution concepts for our bargaining game. We posit that such a solution concept should be *intuitive*, *computationally tractable*, and *unique*. Building on prior work, we offer the set of stable solutions as an intuitive solution concept and provide a complete characterization of this set based on a linear-programming interpretation of the bargaining game. All missing proofs are in the full version of this extended abstract [3].

Theorem 1. *The set of all stable solutions to the network bargaining game can be constructed in polynomial time.*

The set of stable solutions in our game might be quite large, and so, following prior work, we propose balanced solutions as a refinement. We first study the relationship between the stable/balanced concepts and the core/kernel concepts from cooperative game theory. We find that, while these solutions may differ in general, for the constrained bargaining game they exactly coincide. This provides additional motivation for these solution concepts and additionally gives us computational insights from the cooperative game theory literature.

Theorem 2. *An outcome* $\{x_i\}$ *of the constrained bipartite bargaining game is in the core if and only if it corresponds to a stable solution* $(\{z_{ij}\}, M)$*. That is,* $\{x_i\}$ *is in the core if and only if there exists a stable solution* $(\{z_{ij}\}, M)$ *such that* $x_i = \sum_j z_{ij}$ *for all agents* i*.*

Theorem 3. *An outcome* $\{x_i\}$ *of the constrained bipartite bargaining game is in the core intersect prekernel if and only if it corresponds to a balanced solution* $(\{z_{ij}\}, M)$*. That is,* $\{x_i\}$ *is in the core intersect prekernel if and only if there exists a balanced solution* $(\{z_{ij}\}, M)$ *such that* $x_i = \sum_j z_{ij}$ *for all agents* i*.*

Our proofs of theorems 1, 2 and 3 can also be adapted to the matching game studied by Kleinberg and Tardos [28], enabling us to recover some of their results using simple and well-known cooperative game-theoretic constructs.

Using the work of Faigle, Kern, and Kuipers [19], Theorem 3 implies an algorithm for computing *some* balanced solution in constrained bipartite bargaining games (as well as the matching game studied by Kleinberg and Tardos [28]).

Theorem 4. *There is a local dynamics*[3] *that converges to an intersection of prekernel and core, which is a stable, balanced solution of the KT game.*

However, the set of balanced solutions can be quite large, and not all balanced solutions are intuitive. A unique and intuitive balanced solution is the nucleolus, motivated by its symmetry properties. Our final result is an algorithm for computing the nucleolus in constrained bipartite bargaining games.

Theorem 5. *The nucleolus of the constrained bipartite bargaining game can be computed efficiently.*

[3] In fact, the resulting algorithm is similar to the local dynamics proposed in Azar et al. [1].

3 Characterizing Solution Concepts

In this section, we study the solution concepts posed in Section 2 for bipartite graphs with arbitrary capacity constraints. We first define a polytope characterizing all stable solutions (even for non-bipartite graphs). This demonstrates that stable solutions can be computed efficiently and hence helps us understand likely outcomes of our game. We also use our characterization to illustrate the connection between the sets of stable/balanced solutions and the core/kernel of the corresponding cooperative game. This allows us to compute a balanced solution by leveraging existing algorithms for finding a point in the kernel of a cooperative game.

3.1 Characterizing Stable Solutions

We begin by characterizing all stable solutions in the general network bargaining game. In order to do this, a natural approach would be to make "ghost" copies of each node, thereby transforming the general case to the matching case. We demonstrate in the full version that this approach does not work even in the constrained bipartite case. Thus, we are unable to use the existing poset-based characterization of Kleinberg and Tardos [28] for the matching case to solve the general case. Instead, we define a linear program describing the set of optimal contracts and its dual, and use these to characterize the stable solutions, thereby proving Theorem 1. Our linear program is a generalization of the one used by Shapley and Shubik [36] to describe the core for the simpler matching version of the network bargaining game. The optimal contracts can be described by the following linear program:

$$\begin{array}{lll} \text{maximize} & \sum_{ij} w_{ij} x_{ij} & \\ \text{subject to} & \sum_j x_{ij} \leq c_i & \forall i \in N \\ & 0 \leq x_{ij} \leq 1 & \forall (i,j) \in E(G), \end{array} \tag{LP1}$$

where there is a variable x_{ij} for every edge $(i,j) \in E$. The dual of LP1 is:

$$\begin{array}{lll} \text{minimize} & \sum_i u_i c_i + \sum_{ij} y_{ij} & \\ \text{subject to} & u_i + u_j + y_{ij} \geq w_{ij} & \forall (i,j) \in E(G) \\ & 0 \leq y_{ij} \leq 1 & \forall (i,j) \in E(G) \\ & 0 \leq u_i \leq 1 & \forall i \in N. \end{array} \tag{LP2}$$

Given an optimal pair of solutions to these LPs, we show how to construct a stable solution. The primal variables indicate the set of optimal contracts M. We use the dual solution to divide the surplus w_{ij} of the contracts in M. For each contract $(i,j) \in M$, we give u_i to i, u_j to j, and then divide arbitrarily the remaining y_{ij}. Thus $z_{ij} = u_i + \alpha_{ij} y_{ij}$ and $z_{ji} = u_j + (1 - \alpha_{ij}) y_{ij}$ for an arbitrary $\alpha_{ij} \in [0,1]$ (different α_{ij} yield different stable solutions). Conversely, to convert any stable solution to a pair of optimal solutions, we set the primal variables based on the contract M. We define the dual variables as follows: (1) for every unsaturated vertex i, set $u_i = 0$; (2) for every saturated vertex i, set $u_i = \min_{j \in N_i} z_{ij}$; (3) for every $x_{ij} = 0$, set $y_{ij} = 0$; (4) for every $x_{ij} = 1$, set $y_{ij} = w_{ij} - u_i - u_j$. To prove that these constructions work, we use the complementary slackness conditions.

3.2 Relating Bargaining Solutions to Cooperative Game Theory Solutions

Whereas the notions of stable and balanced solutions have been only recently introduced, the cooperative game theoretic notions of core and kernel are well-studied. Hence it is interesting to relate these two seemingly different notions. There is no general reason why we would expect these notions to coincide, and in fact, as we demonstrate via an example, they do not for the general bipartite bargaining case. Nonetheless, for the constrained bipartite bargaining game, we are able to prove that stable/balanced and core/kernel do coincide. In the below discussions, we say outcome $\{x_i\}$ has been produced by solution $(\{z_{ij}\}, M)$, if x_i is equal to total amount of money which person i earns in solution $(\{z_{ij}\}, M)$.

Stable = Core. First we prove that an outcome $\{x_i\}$ produced by any stable solution $(\{z_{ij}\}, M)$ is in the core. Based on the discussion in Section 3.1, it is enough to prove that for every integer optimum solution of LP1 and every real optimum solution of LP2, all outcomes produced by these solutions are in the core.

Second we prove that for every $\{x_i\}$, there is a stable solution $(\{z_{ij}\}, M)$ which produces it. In the constrained bipartite graph $c_j = 1$ for every $j \in V_2$, and thus given the set of contracts M, the $\{z_{ij}\}$ are uniquely determined by the core outcome $\{x_i\}$ (for every $(i, j) \in M$ with $j \in V_2$ we set $z_{ji} = x_j$ and $z_{ij} = w_{ij} - x_j$). The crux of the problem is therefore in choosing M and using the core inequalities to prove that the resulting solution is stable.

Balanced = Kernel. Next, we prove that Stable ∩ Balanced is equivalent to Core ∩ Kernel. From the discussion above, we can assume we are provided with a stable solution $(\{z_{ij}\}, M)$ and associated core outcome $\{x_i\}$. The goal is to show that the solution is balanced if and only if the outcome is in the kernel.

Recall that the outcome $\{x_i\}$ is in kernel if and only if $s_{ij} = s_{ji}$ for all pairs of players i, j. On the other hand, a solution is balanced if and only if for any $(i, j) \in M$, $z_{ij} - \alpha_i = z_{ji} - \alpha_j$. We define the net gain of player i after losing the contract with j as $\tilde{s}_{ij} := \alpha_i - z_{ij}$ and prove that $\tilde{s}_{ij} = s_{ij}$ for all edges $(i, j) \in M$. This finishes the proof.

4 Finding the Nucleolus

In this section we propose an algorithm to find the nucleolus for our constrained bipartite bargaining game in polynomial time. Previous works [27,40,19] show how to compute the nucleolus via an iterated LP-based algorithm, and we also adopt this approach. The main complication in applying this algorithm is that the natural LP does not have polynomial size, and hence we must prune it without sacrificing the correctness of the algorithm.

We describe this general approach as it applies to our setting. As the core is not empty in our constrained bipartite bargaining game (see Section 3), the nucleolus will be in the core. Hence, our task is to search for a core outcome $\{x_i\}$ which maximizes the lexicographically-sorted sequence of set excesses $\epsilon = (\epsilon_{S_1}, \ldots, \epsilon_{S_{2^N}})$; see Definition 7. We proceed iteratively. In each iteration, we search for the "next" element ϵ_{S_i}

of ϵ by solving a linear program whose objective defines ϵ_{S_i}. In doing so, we must be careful to constrain certain variables appropriately such that the computations of previous iterations carry through. As there are exponentially many elements of the vector ϵ, we cannot simply introduce an equality for each element that we wish to fix. Rather, we show how to construct a polynomially-sized set of "representatives" and argue that these suffice to describe the excesses of sets fixed in all previous iterations.

References

1. Azar, Y., Birnbaum, B., Celis, L.E., Devanur, N.R., Peres, Y.: Convergence of local dynamics to balanced outcomes in exchange networks. In: FOCS (2009)
2. Azar, Y., Devanur, N.R., Jain, K., Peres, Y.: Monotonicity in bargaining games. In: SODA (2010)
3. Bateni, M., Hajiaghayi, M., Immorlica, N., Mahini, H.: The cooperative game theory foundations of network bargaining games, CoRR, abs/1004.4317 (2010)
4. Bondareva, O.N.: Some applications of linear programming to cooperative games. Problemy Kibernetiki 10, 119–139 (1963)
5. Bosse, H., Byrka, J., Markakis, E.: New algorithms for approximate nash equilibria in bimatrix games. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 17–29. Springer, Heidelberg (2007)
6. Chakraborty, T., Kearns, M.: Bargaining solutions in a social network. In: Papadimitriou, C., Zhang, S. (eds.) WINE 2008. LNCS, vol. 5385, pp. 548–555. Springer, Heidelberg (2008)
7. Chakraborty, T., Kearns, M., Khanna, S.: Network bargaining: Algorithms and structural results. In: EC (2009)
8. Charness, G., Corominas-Bosch, M., Frechette, G.R.: Bargaining and Network Structure: An Experiment, SSRN eLibrary (2005)
9. Chen, X., Deng, X.: Settling the complexity of two-player nash equilibrium. In: FOCS, pp. 261–272 (2006)
10. Chen, X., Deng, X., Teng, S.-H.: Computing nash equilibria: Approximation and smoothed complexity. In: FOCS 2006: 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 603–612. IEEE Computer Society, Los Alamitos (2006)
11. Conitzer, V., Sandholm, T.: Complexity of determining nonemptiness of the core. In: EC, pp. 230–231 (2003)
12. Daskalakis, C., Mehta, A., Papadimitriou, C.H.: A note on approximate nash equilibria. In: Spirakis, P.G., Mavronicolas, M., Kontogiannis, S.C. (eds.) WINE 2006. LNCS, vol. 4286, pp. 297–306. Springer, Heidelberg (2006)
13. Daskalakis, C., Papadimitriou, C.H.: On oblivious ptas's for nash equilibrium. In: STOC, pp. 75–84 (2009)
14. Davis, M., Maschler, M.: The kernel of a cooperative game. Naval Research Logistics Quarterly 12, 223–259 (1965)
15. Deng, X., Fang, Q., Sun, X.: Finding nucleolus of flow game. In: SODA, pp. 124–131 (2006)
16. Driessen, T.S.H.: Cooperative Games: Solutions and Applications. Kluwer Academic Publishers, Dordrecht (1988)
17. Faigle, U., Kern, W., Fekete, S.P., Hochstättler, W.: The nucleon of cooperative games and an algorithm for matching games. Mathematical Programming 83, 195–211 (1998)
18. Faigle, U., Kern, W., Kuipers, J.: Computing the nucleolus of min-cost spanning tree games is np-hard. International Journal of Game Theory 27, 443–450 (1998)
19. Faigle, U., Kern, W., Kuipers, J.: An efficient algorithm for nucleolus and prekernel computation in some classes of TU-games, Memorandum 1464, University of Twente, Enschede (1998)

20. Faigle, U., Kern, W., Kuipers, J.: On the computation of the nucleolus of a cooperative game. International Journal of Game Theory 30, 79–98 (2001)
21. Faigle, U., Kern, W., Kuipers, J.: Computing an element in the lexicographic kernel of a game. Mathematical methods of operations research 63, 427–433 (2006)
22. Gilies, D.B.: Solutions to general non-zero-sum games. Ann. Math. Studies 40, 47–85 (1959)
23. Granot, D., Maschler, M., Owen, G., Zhu, W.R.: The kernel/nucleolus of a standard tree game. International Journal of Game Theory 25(2), 219–244 (1996)
24. Immorlica, N., Mahdian, M., Mirrokni, V.S.: Limitations of cross-monotonic cost sharing schemes. In: SODA, pp. 602–611 (2005)
25. Jain, K., Vazirani, V.: Applications of approximation algorithms to cooperative games. In: STOC, pp. 364–372 (2001)
26. Kanoria, Y., Bayati, M., Borgs, C., Chayes, J.T., Montanari, A.: A natural dynamics for bargaining on exchange networks, CoRR, abs/0911.1767 (2009)
27. Kern, W., Paulusma, D.: Matching games: the least core and the nucleolus. Math. Oper. Res. 28(2), 294–308 (2003)
28. Kleinberg, J., Tardos, É.: Balanced outcomes in social exchange networks. In: STOC, pp. 295–304 (2008)
29. Kontogiannis, S.C., Spirakis, P.G.: Efficient algorithms for constant well supported approximate equilibria in bimatrix games. In: ICALP, pp. 595–606 (2007)
30. Markakis, E., Saberi, A.: On the core of the multicommodity flow game. In: EC, pp. 93–97 (2003)
31. Meinhardt, H.: An lp approach to compute the pre-kernel for cooperative games. Computers & Operations Research 33, 535–557 (2006)
32. Nash, J.F.: The bargaining problem. Econometrica 18, 155–162 (1950)
33. Neumann, J.V., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton (1944)
34. Potters, J.A.M.: An axiomatization of the nucleolus. International Journal of Game Theory 19(4), 365–373 (1991)
35. Schmeidler, D.: The nucleolus of a characteristic function game. SIAM Journal of Applied Mathematics 17(6), 1163–1170 (1969)
36. Shapley, L.S., Shubik, M.: The assignment game i: the core. International Journal of Game Theory, 111–130 (1972)
37. Shapley, L.S.: On balanced sets and cores. Naval Research Logistics Quarterly 14, 453–460 (1967)
38. Snijders, C.: Axiomatization of the nucleolus. Math. Oper. Res. 20(1), 189–196 (1995)
39. Sobolev, A.: The nucleolus for cooperative games with arbitrary bounds of individual rationality. International Journal of Game Theory 24, 13–22 (1995)
40. Solymosi, T., Raghavan, T.E.S.: An algorithm for finding the nucleolus of assignment games. International Journal of Game Theory 23, 119–143 (1994)
41. Stearns, R.E.: Convergent transfer schemes for n-person games. Transactions of American Mathematical Society 134, 449–459 (1968)
42. Tsaknakis, H., Spirakis, P.G.: An optimization approach for approximate nash equilibria. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 42–56. Springer, Heidelberg (2007)
43. West, D.B.: Introduction to Graph Theory, 2nd edn. Prentice-Hall, Englewood Cliffs (2000)
44. Yarom, M.: The lexicographic kernel of a cooperative game. Math. Oper. Res. 6, 66–100 (1981)

On the Existence of Pure Nash Equilibria in Weighted Congestion Games

Tobias Harks* and Max Klimm**

Department of Mathematics, TU Berlin, Germany
{harks,klimm}@math.tu-berlin.de

Abstract. We study the existence of pure Nash equilibria in weighted congestion games. Let C denote a set of cost functions. We say that C is *consistent* if every weighted congestion game with cost functions in C possesses a pure Nash equilibrium. We say that C is *FIP-consistent* if every weighted congestion game with cost functions in C has the Finite Improvement Property. Our main results are structural characterizations of consistency for twice continuously differentiable cost functions. More specifically, we show that C is consistent for two-player games if and only if C contains only monotonic functions and for all $c_1, c_2 \in C$, there are constants $a, b \in \mathbb{R}$ such that $c_1 = a\,c_2 + b$. For games with at least 3 players we show that C is consistent if and only if exactly one of the following cases hold: (*i*) C contains only affine functions; (*ii*) C contains only exponential functions such that $c(\ell) = a_c\, e^{\phi \ell} + b_c$ for some $a_c, b_c, \phi \in \mathbb{R}$, where a_c and b_c may depend on c, while ϕ must be equal for every $c \in C$. This characterization is even valid for 3-player games, thus, closing the gap to 2-player games considered above. Finally, we derive various results regarding consistency and FIP-consistency for weighted network congestion games.

1 Introduction

In many situations, the state of a system is determined by a finite number of independent players, each optimizing an individual objective function. A natural framework for analyzing such decentralized systems are noncooperative games. While it is well known that for finite noncooperative games an equilibrium point in mixed strategies always exists, this need not be true for Nash equilibria in pure strategies (PNE for short). One of the fundamental goals in algorithmic game theory is to characterize conditions under which a Nash equilibrium in pure strategies exists. In this paper, we study this question for weighted congestion games.

Congestion games, as introduced by Rosenthal [18], model the interaction of a finite set of strategic agents that compete over a finite set of facilities. A pure strategy of each player is a set of facilities. The cost of facility f is given by a real-valued cost function c_f that depends on the number of players using f and the private cost of every player

* Research supported by the Federal Ministry of Education and Research (BMBF grant 03MOPAI1).

** Research supported by the Deutsche Forschungsgemeinschaft within the research training group 'Methods for Discrete Structures' (GRK 1408).

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 79–89, 2010.

equals the sum of the costs of the facilities in the strategy that she chooses. Rosenthal [18] proved in a seminal paper that such congestion games always admit a PNE by showing these games posses a potential function[1]. In a weighted congestion game, every player has a demand $d_i \in \mathbb{R}_{>0}$ that she places on the chosen facilities. The cost of a facility is a function of the total load on the facility. An important subclass of weighted congestion games are weighted *network* congestion games. Every player is associated with a positive demand that she wants to route from her origin to her destination on a path of minimum cost. In contrast to unweighted congestion games, weighted congestion games do not always admit a PNE. Fotakis et al. [8] and Libman and Orda [15] constructed a single-commodity network instance with two players having demands one and two, respectively. Their instances use different non-decreasing cost values per edge that are defined at the three possible loads $1, 2, 3$. Goemans et al. [11] constructed a two-player single-commodity instance without a PNE that uses different polynomial cost functions with nonnegative coefficients and degree of at most two. On the positive side, Fotakis et al. [8,9] proved that every weighted congestion game with affine cost functions possesses a PNE. While the negative results of [8,11,15] suggest that only affine functions guarantee the existence of PNE, a result of Panagopoulou and Spirakis [17] came as a surprise: PNE always exist for instances with exponential cost functions. It is worth noting that the positive results of [8,9,17] are particularly important as they establish existence of PNE for the respective sets of cost functions *independent* of the underlying game structure, that is, *independent* of the underlying strategy set, the demand vector and the number of players, respectively.

In this paper, we further explore the equilibrium existence problem. Our goal is to precisely characterize, which type of cost functions actually guarantees the existence of PNE. To formally capture this issue, we introduce the notion of *PNE-consistency* or simply *consistency* of a set of cost functions. Let $\mathcal{C}$ be a set of cost functions and let $\mathcal{G}(\mathcal{C})$ be the set of *all* weighted congestion games with cost functions in $\mathcal{C}$. We say that $\mathcal{C}$ is *consistent* if every game in $\mathcal{G}(\mathcal{C})$ possesses a PNE. Using this terminology, the results of [8,9,17] yield that $\mathcal{C}$ is consistent if $\mathcal{C}$ contains either affine or exponential functions.

A natural open question is to decide whether there are further consistent functions, that is, functions guaranteeing the existence of a PNE. We thus investigate the following question: How large is the set $\mathcal{C}$ of consistent cost functions?

1.1 Our Results

We give an almost complete characterization of consistency of cost functions in weighted congestion games. Specifically, we show the following: Let $\mathcal{C}$ be a nonempty set of continuous functions.

1. If $\mathcal{C}$ is consistent, then $\mathcal{C}$ may only contain monotonic functions. We further show that monotonicity of cost functions is necessary for consistency even in singleton games, two-player games, two facility games, games with identical cost functions and games with symmetric strategies.

[1] A real-valued function P on the set of strategies having the property that every improving move by one defecting player strictly reduces the value of P.

2. Let C be a nonempty set of twice continuously differentiable functions. Then, C is consistent for two-player games if and only if C only contains only monotonic functions and for all $c_1, c_2 \in C$, there are constants $a, b \in \mathbb{R}$ such that $c_1 = a\,c_2 + b$. To the best of our knowledge, it was not known before that two-player games possess PNE for the above class of cost functions.
3. Let C be a nonempty set of twice continuously differentiable functions. We prove that C is consistent for games with at least 3 players if and only if exactly one of the following cases hold: (*i*) C contains only affine functions; (*ii*) C contains only exponential functions such that $c(\ell) = a_c\, e^{\phi\,\ell} + b_c$ for some $a_c, b_c, \phi \in \mathbb{R}$, where a_c and b_c may depend on c, while ϕ must be equal for every $c \in C$. This characterization is even valid for 3-player games, thus, closing the gap to 2-player games considered above. We note that the "if" direction follows from [8,9,17].
4. Finally, we study weighted network congestion games. Let C be a non-empty set of strictly increasing, positive and twice continuously differentiable functions. For multi-commodity networks with at least three players, C is consistent if and only if C contains only affine functions or certain exponential functions as specified above. For two-player network games (single or two-commodity networks), we show that C is consistent if and only if for all $c_1, c_2 \in C$, there are constants $a, b \in \mathbb{R}$ such that $c_1 = a\,c_2 + b$. This characterization gives a structural insight, why the instances used in [8,11,15] do not possess a PNE.

   For single-commodity network games with at least three players, we prove that C is FIP-consistent if and only if C contains only affine functions or certain exponential functions.

All proofs missing in the extended abstract are presented in the full version [12].

1.2 Significance and Techniques

Weighted congestion games are one of the core topics in the algorithmic game theory, operations research, and economics literature. In particular, there is a large body of work quantifying the price of anarchy of PNE for different sets of cost functions, see Awerbuch et al. [4], Christodoulou and Koutsoupias [5], and Aland et al. [2]. There are, however, drawbacks in the use of Nash equilibria. While mixed Nash equilibria are guaranteed to exist, their use is unrealistic in many games. On the other hand, pure Nash equilibria as the stronger solution concept may fail to exist in weighted congestion games. Thus, we believe that our work constitutes an important step towards fully characterizing the PNE-existence problem in weighted congestion games.

Our characterizations essentially rely on two ingredients. First, we derive in Section 3 for continuous and consistent cost functions two necessary conditions (Monotonicity Lemma and Extended Monotonicity Lemma). The Monotonicity Lemma states that any continuous and consistent cost function must be monotonic. The Lemma is proved by constructing a sequence of two-player weighted congestion games in which we identify a unique 4-cycle of deviations of two players. Then, we show that for any non-monotonic cost function, there is a weighted congestion game with a unique improvement cycle. By adding additional players and carefully choosing player-weights

and strategy spaces, we then derive the Extended Monotonicity Lemma, which ensures that the set of cost functions contained in a certain *finite integer linear hull* of the considered cost functions must be monotone. The proof of the Extended Monotonicity Lemma provides a template for constructing an instance without a PNE, for any non-affine and non-exponential cost function; we illustrate this template by constructing an instance without PNE using identical cubic cost functions.

In Section 4, we give a characterization of the set of functions that arise from affine transformations of a monotonic function and show that the Extended Monotonicity Lemma for two-player games implies that consistent cost functions must be of this form. In Section 5, we characterize the set of affine and exponential functions, and show that the Extended Monotonicity Lemma for games with at least three players implies that consistent cost functions must be either affine or exponential. In Section 6, we discuss implications of the Extended Monotonicity Lemma when applied to weighted network congestion games.

1.3 Related Work

Milchtaich [16] showed that singleton weighted congestion games with player-specific cost functions do not always have a PNE. Milchtaich further characterized topological properties of networks that guarantee the existence of PNE in network congestion games if players control different amounts of flow or cost functions are player specific. This was further elaborated by Gairing et al. [10], who considered different cost functions, slight modifications in the network topology, and both the weighted and the unweighted cases. Harks et al. [13] proved that every weighted congestion game with two-players and uniform (up to affine translations) and strictly monotonic cost functions possesses a PNE.

Ieong et al. [14] proved that in singleton congestion games with non-decreasing cost functions, best response dynamics converge in polynomial time to a PNE. Ackermann et al. [1] extended this result to weighted congestion games with a so called *matroid property*, that is, the strategy of every player forms a basis of a matroid. In the same paper, they also characterized the existence of PNE in weighted congestion games with the same matroid property by proving that for any strategy space not satisfying the matroid property, there is an instance of a weighted congestion game not having a PNE. Fanelli and Moscardelli [7] studied convergence properties of certain improvement dynamics in weighted congestion games with polynomial cost functions of bounded degree. Dunkel and Schulz [6] proved that it is strongly NP-hard to decide whether or not a weighted congestion game with nonlinear cost functions possesses a PNE.

2 Preliminaries

We consider finite strategic games $G = (N, X, \pi)$, where $N = \{1, \ldots, n\}$ is the non-empty and finite set of players, $X = \bigtimes_{i \in N} X_i$ is the non-empty strategy space, and $\pi : X \to \mathbb{R}^n$ is the combined *private cost* function that assigns a private cost vector $\pi(x)$ to each strategy profile $x \in X$. These games are cost minimization games. Unless specified

otherwise, we allow private cost functions to be negative or positive. A strategic game is called *finite* if X is finite.

We use standard game theory notation; for a player $i \in N$ we denote the set $N \setminus \{i\}$ by $-i$. A strategy profile x is a pure Nash equilibrium if there is no player $i \in N$ having an alternative strategy profile $y_i \in X_i$ such that $\pi_i(y_i, x_{-i}) - \pi_i(x) < 0$.

A tuple $\mathcal{M} = (N, F, X = \bigtimes_{i \in N} X_i, (c_f)_{f \in F})$ is called a *congestion model*, where F is a non-empty, finite set of facilities, for each player $i \in N$, her collection of pure strategies X_i is a non-empty, finite set of subsets of F and $(c_f)_{f \in F}$ is a set of cost functions. In the following, we will define weighted congestion games similar to Goemans et al. [11]. Let $\mathcal{M} = (N, F, X, (c_f)_{f \in F})$ be a congestion model and $(d_i)_{i \in N}$ be a vector of demands with $d_i \in \mathbb{R}_{>0}$. The corresponding *weighted congestion game* is the game $G(\mathcal{M}) = (N, X, \pi)$, where π is defined as $\pi_i(x) = \sum_{f \in x_i} d_i c_f(\ell_f(x))$ and $\ell_f(x) = \sum_{j \in N : f \in x_j} d_j$ is called the *load* on facility f in strategy x. We will write G as shorthand for $G(\mathcal{M})$. Now, we will define the notion of *consistency* of cost functions. Let C be a set of cost functions and let $\mathcal{G}(C)$ be the set of all weighted congestion games with cost functions in C. Then, C is *consistent* if every $G \in \mathcal{G}(C)$ possesses a PNE. If the set $\mathcal{G}(C)$ is restricted, e.g., two player games etc., we say that C is *consistent w.r.t.* $\mathcal{G}(C)$ if every $G \in \mathcal{G}(C)$ possesses a PNE. A pair $(x, (y_i, x_{-i})) \in X \times X$ is called an *improving move* of player i if $\pi_i(x_i, x_{-i}) - \pi_i(y_i, x_{-i}) > 0$. We denote by I the set of improving moves in G. We call a sequence $\gamma = (x^0, x^1, \ldots)$ an *improvement path* if $(x^k, x^{k+1}) \in I$ for all k. If every improvement path is finite, G has the *Finite Improvement Property (FIP)*. We say that C is *FIP-consistent*, if every $G \in \mathcal{G}(C)$ has the FIP.

3 Necessary Conditions on the Existence of a PNE

Throughout this work, we will assume that C is a set of continuous functions. As a first result, we prove that if C is consistent, then every function $c \in C$ is monotonic. We will first need a (nontrivial) technical lemma, which will be used many times.

Lemma 1. *Let $c : \mathbb{R}_{\geq 0} \to \mathbb{R}$ be a continuous function. Then, the following two statements are equivalent:*

1. *c is monotonic on $\mathbb{R}_{\geq 0}$.*
2. *The following two conditions hold:*
   (a) *For all $x \in \mathbb{R}_{>0}$ with $c(x) < c(0)$ there is $\varepsilon > 0$ such that $c(y) \leq c(x)$ for all $y \in (x, x + \varepsilon)$,*
   (b) *For all $x \in \mathbb{R}_{>0}$ with $c(x) > c(0)$ there is $\varepsilon > 0$ such that $c(y) \geq c(x)$ for all $y \in (x, x + \varepsilon)$.*

We proceed by presenting a necessary condition for consistency.

Lemma 2 (Monotonicity Lemma). *Let C be a set of continuous functions. If C is consistent, then every $c \in C$ is monotonic on $\mathbb{R}_{>0}$.*

Besides the continuity of the functions in C, the proof of Lemma 2 relies on rather mild assumptions and, thus, this result can be strengthened in the following way.

Corollary 1. *Let C be a set of continuous functions. Let $\mathcal{G}(C)$ be the set of weighted congestion games with cost functions in C satisfying one or more of the following properties: (i) Each game $G \in \mathcal{G}(C)$ has two players; (ii) Each game $G \in \mathcal{G}(C)$ has two facilities; (iii) For each game $G \in \mathcal{G}(C)$ and each player $i \in N$, the set of her strategies X_i contains a single facility only; (iv) Each game $G \in \mathcal{G}(C)$ has symmetric strategies; (v) Cost functions are identical, that is, $c_f = c_g$ for all $f, g \in F$. If C is consistent w.r.t. $\mathcal{G}(C)$, then, each $c \in C$ must be monotonic.*

In particular, if there is a non-monotonic function $\tilde{c} \in C$, then, there is a symmetric singleton weighted congestion game with identical cost functions, two facilities, two players that does not admit a PNE. However, it is well known that singleton games posses a PNE if cost functions are non-decreasing.

We now extend the Monotonicity Lemma to obtain a stronger necessary condition by regarding more players and more complex strategies. To this end, we consider for finite K those functions that can be written as the sum of K functions in C, possibly with an offset. Formally, we define the *finite integer linear hull* of C as

$$\mathcal{L}_{\mathbb{Z}}(C) = \{c : \mathbb{R}_{\geq 0} \to \mathbb{R} : c(x) = \sum_{k=1}^{K} a_k c_k(x + b_k), K \in \mathbb{N}, a_k \in \mathbb{Z}, b_k \geq 0, c_k \in C\}, \quad (1)$$

and show that consistency of C implies that $\mathcal{L}_{\mathbb{Z}}(C)$ contains only monotonic functions.

Lemma 3 (Extended Monotonicity Lemma). *Let C be a set of continuous functions. If C is consistent, then $\mathcal{L}_{\mathbb{Z}}(C)$ contains only monotonic functions.*

Proof. Let $\tilde{c} \in \mathcal{L}_{\mathbb{Z}}(C)$ be arbitrary. By allowing $c_k = c_l$ for $k \neq l$, we can omit the integer coefficients a_k and rewrite $\tilde{c}$ as $\tilde{c}(x) = \sum_{k=1}^{m_+} c_k(x + b_k) - \sum_{k=1}^{m_-} \bar{c}_k(x + \bar{b}_k)$ for some $c_k, \bar{c}_k \in C, m_+, m_- \in \mathbb{N}$.

We define a congestion model $\mathcal{M} = (N, F, X, (c_f)_{f \in F})$, with $N = N_p \cup N^+ \cup N^-$ and $F = F^1 \cup F^2 \cup F^3 \cup F^4$. The set of players N^+ contains for each c_k, $1 \leq k \leq m_+$, a player with demand b_k and the set of players N^- contains for each $\bar{c}_k$, $1 \leq k \leq m_-$, a player with demand $\bar{b}_k$. We call the players in $N^- \cup N^+$ *offset* players. The set $N_p = \{1, 2\}$ contains two additional (non-trivial) players.

We now explain the strategy spaces and the sets F^1, F^2, F^3, F^4. For each function c_k, $1 \leq k \leq m_+$, we introduce two facilities f_k^2, f_k^3 with cost function c_k. For each function $\bar{c}_k$, $1 \leq k \leq m_-$, we introduce two facilities f_k^1, f_k^4 with cost function $\bar{c}_k$. To model the offsets b_k in (1), for each offset player $k \in N^+$, we define $X_k = \{\{f_k^2, f_k^3\}\}$. Similarly, for each offset player $k \in N^-$, we set $X_k = \{\{f_k^1, f_k^4\}\}$. The non-trivial players in N_p have strategies $X_1 = \{F^1 \cup F^2, F^3 \cup F^4\}$ and $X_2 = \{F^1 \cup F^3, F^2 \cup F^4\}$, where $F^1 = \{f_1^1, \ldots, f_{m_-}^1\}, F^2 = \{f_1^2, \ldots, f_{m_+}^2\}$,
$F^3 = \{f_1^3, \ldots, f_{m_+}^3\}$, and $F^4 = \{f_1^4, \ldots, f_{m_-}^4\}$. We consider a family of weighted congestion games $G_\delta^x(\mathcal{M})$ parameterized by $d_1 = \delta$ and $d_2 = x$ for $1, 2 \in N_p$. For the 4-cycle

$$\gamma = \Big((F^1 \cup F^2, F^1 \cup F^3, \ldots), (F^3 \cup F^4, F^1 \cup F^3, \ldots), (F^3 \cup F^4, F^2 \cup F^4, \ldots), \\ (F^1 \cup F^2, F^2 \cup F^4, \ldots), (F^1 \cup F^2, F^1 \cup F^3, \ldots)\Big),$$

we calculate

$$\begin{aligned}
&\pi_1(F^3 \cup F^4, F^1 \cup F^3, \dots) - \pi_1(F^1 \cup F^2, F^1 \cup F^3, \dots) \\
&= d_1 \sum_{k=1}^{m_+} c_k(d_1 + d_2 + b_k) - d_1 \sum_{k=1}^{m_-} \bar{c}_k(d_1 + d_2 + \bar{b}_k) + d_1 \sum_{k=1}^{m_-} \bar{c}_k(d_1 + \bar{b}_k) - d_1 \sum_{k=1}^{m_+} c_k(d_1 + b_k) \\
&= d_1(\tilde{c}(x+\delta) - \tilde{c}(\delta)). \qquad (2)
\end{aligned}$$

Similarly, we obtain

$$\begin{aligned}
\pi_2(F^3 \cup F^4, F^2 \cup F^4, \dots) - \pi_2(F^3 \cup F^4, F^1 \cup F^3, \dots) &= d_2(\tilde{c}(x) - \tilde{c}(x+\delta)) \\
\pi_1(F^1 \cup F^2, F^2 \cup F^4, \dots) - \pi_1(F^3 \cup F^4, F^2 \cup F^4, \dots) &= d_1(\tilde{c}(x+\delta)) - \tilde{c}(\delta) \qquad (3) \\
\pi_2(F^1 \cup F^2, F^1 \cup F^3, \dots) - \pi_2(F^1 \cup F^2, F^2 \cup F^4, \dots) &= d_2(\tilde{c}(x) - \tilde{c}(x+\delta)).
\end{aligned}$$

If $\tilde{c}(x) > \tilde{c}(0)$, we can find $\varepsilon > 0$ such that $\tilde{c}(x+\delta) > \tilde{c}(\delta)$ for all $0 < \delta < \varepsilon$. For such δ, the values in (2) and (3) are negative and we may conclude that there is an improvement cycle in the game G if $\tilde{c}(x+\delta) - \tilde{c}(x) < 0$. Hence, we have $\tilde{c}(y) - \tilde{c}(x) \geq 0$ for all $y \in (x, x+\varepsilon)$. If $\tilde{c}(0) > \tilde{c}(x)$, considering the 4-cycle in the other direction yields the claimed result by the same argumentation. Using that every strategy combination is contained in γ and applying Lemma 1 delivers the claimed result. □

4 A Characterization for Two-Player Games

We will analyze implications of the Extended Monotonicity Lemma (Lemma 3) for two-player weighted congestion games. First, we remark that if all offsets b_k and $\bar{b}_k$ in (1) are equal to zero, the construction in Lemma 3 only involves two players. For ease of exposition, we additionally restrict ourselves to the case $K = 2$, that is, we only regard those functions that can be written as the difference of two functions in C without offset. Formally, define

$$\mathcal{L}^2_{\mathbb{N}}(C) = \{c : \mathbb{R}_{\geq 0} \to \mathbb{R} \;:\; c(x) = a_1 c_1(x) - a_2 c_2(x), a_1, a_2 \in \mathbb{N}, c_1, c_2 \in C\}.$$

Then, we obtain the following immediate corollary of the Extended Monotonicity Lemma.

Corollary 2. *Let C be a set of continuous functions. If C is consistent w.r.t. two-player games, then, $\mathcal{L}^2_{\mathbb{N}}(C)$ contains only monotonic functions.*

Thus, we will proceed investigating sets of functions C that guarantee that $\mathcal{L}^2_{\mathbb{N}}(C)$ contains only monotonic functions. For this purpose, we first need the following technical lemma.

Lemma 4. *Let C be a set of functions that are twice continuously differentiable. Then, the following are equivalent:*

1. *$\mathcal{L}^2_{\mathbb{N}}(C)$ contains only monotonic functions.*
2. *For all $c_1, c_2 \in C$ there are $a, b \in \mathbb{R}$ such that $c_2(x) = a\,c_1(x) + b$ for all $x \geq 0$.*

Using Lemma 4, we are now ready to state our first main result.

Theorem 1. *Let C be a set of twice continuously differentiable functions. Let $\mathcal{G}^2(C)$ be the set of two-player games such that cost functions are in C. Then, the following two conditions are equivalent.*

1. *C is consistent w.r.t. $\mathcal{G}^2(C)$*
2. *C contains only monotonic functions and for all $c_1, c_2 \in C$, there are constants $a, b \in \mathbb{R}$ such that $c_1 = a\,c_2 + b$.*

5 A Characterization for the General Case

We now consider the case $n \geq 3$, that is, we consider weighted congestion games with at least three players. We will show that a set of twice continuously differentiable cost functions is consistent if and only if this set contains either linear or certain exponential functions. Our main tool for proving this result is to analyze implications of the Extended Monotonicity Lemma (Lemma 3) for three-player weighted congestion games. Formally, define

$$\mathcal{L}^3_{\mathbb{N}}(C) = \{c : \mathbb{R}_{\geq 0} \to \mathbb{R} : c(x) = a_1 c_1(x) - a_2 c_2(x+b), a_1, a_2 \in \mathbb{Z}, c_1, c_2 \in C, b \in \mathbb{R}_{>0}\}.$$

Then, we obtain the following immediate corollary of the Extended Monotonicity Lemma.

Corollary 3. *Let C be a set of continuous functions. If C is consistent w.r.t three-player games, then $\mathcal{L}^3_{\mathbb{N}}(C)$ contains only monotonic functions.*

Note that $\mathcal{L}^3_{\mathbb{N}}(C)$ involves a single offset b, which requires only three players in the construction of the proof of the Extended Monotonicity Lemma.

In order to characterize the sets of functions C such that $\mathcal{L}^3_{\mathbb{N}}(C)$ contains only monotonic functions, we need the following technical lemma.

Lemma 5. *Let c be a twice continuously differentiable function. Then, the following two are equivalent:*

1. *Either $c(\ell) = a\,e^{\phi \ell} + b$ for some $a, b, \phi \in \mathbb{R}$, or $c(\ell) = a\,x + b$ for some $a, b \in \mathbb{R}$.*
2. $\det\begin{pmatrix} c'(x) & c'(y) \\ c''(x) & c''(y) \end{pmatrix} = 0$ *for all $x, y \in \mathbb{R}_{>0}$.*

We are now ready to state our second main theorem.

Theorem 2. *Let C be a set of twice continuously differentiable functions. Then, C is consistent if and only if one of the following cases holds*

1. *C contains only affine functions*
2. *C contains only functions of type $c(\ell) = a_c\,e^{\phi \ell} + b_c$ where $a_c, b_c \in \mathbb{R}$ may depend on c while $\phi \in \mathbb{R}$ is independent of c.*

We conclude this section by giving an example that captures the main ideas presented so far. Theorem 2 establishes that for each non-affine and non-exponential cost function $\tilde{c}$, there is a weighted congestion game G with uniform cost function $\tilde{c}$ on all facilities that does not admit a PNE. In the following example, we show how such a game for $c(\ell) = \ell^3$ is constructed.

Example 1. As $c(\ell) = \ell^3$ is neither affine nor exponential, there are $a_1, a_2 \in \mathbb{N}$ and $x \in \mathbb{R}_{>0}$ such that $\tilde{c}(\ell) = a_1\, c(\ell) - a_2\, c(\ell + x)$ has a strict extremum. In fact, we can choose $a_1 = 2, a_2 = 2$ and $x = 1$, that is, the function $\tilde{c}(\ell) = 2c(\ell) - c(\ell + 1) = 2\ell^3 - (\ell + 1)^3$ has a strict local minimum at $\ell = 1 + \sqrt{2}$. In particular, we can choose $d_1 = 1$ and $d_2 = 2$ such that $\tilde{c}(d_1) = -6 > \tilde{c}(d_2) = -11 < \tilde{c}(d_1 + d_2) = -10$. The weighted congestion game without PNE is now constructed as follows: We introduce $2(a_1 + a_2)$ facilities $f_1, \ldots, f_6$ and the following strategies $x_1^a = \{f_1, f_2, f_3\}$, $x_1^b = \{f_4, f_5, f_6\}$, $x_2^a = \{f_1, f_2, f_4\}$, $x_2^b = \{f_3, f_5.f_6\}$, and $x_3 = \{f_3, f_4\}$. We then set $X_1 = \{x_1^a, x_1^b\}$, $X_2 = \{x_2^a, x_2^b\}$, and $X_3 = \{x_3\}$. The so defined game has four strategy profiles, namely (x_1^a, x_2^a, x_3), (x_1^a, x_2^b, x_3), (x_1^b, x_2^a, x_3), (x_1^b, x_2^b, x_3). As Player 3 is an offset player, she has a single strategy only, thus, the players' private costs depend only on the choice of Players 1 and 2 as indicated in the table in Fig. 1. We derive that the 4-cycle γ depicted in Fig. 1 is a best-reply cycle in G. As there are no strategy profiles outside γ, G has no PNE.

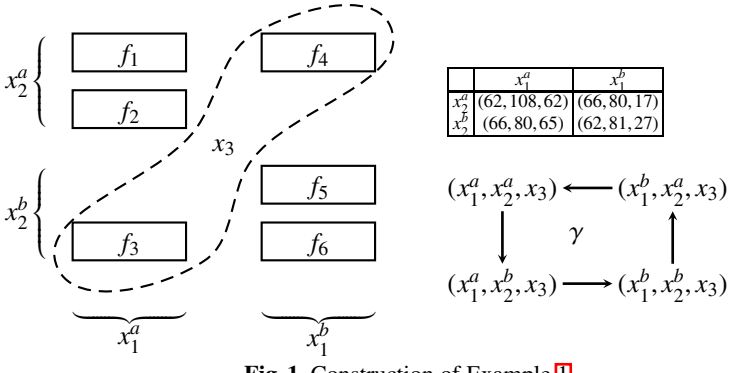


Fig. 1. Construction of Example 1

6 Weighted Network Congestion Games

We discuss the implications of our characterizations to the important subclass of weighted network congestion games. We start with multi-commodity networks.

Multi-commodity Networks. Our characterization of consistent cost functions given in Theorem 2 crucially relies on the construction used in the Extended Monotonicity Lemma. By assuming that cost functions are strictly increasing and positive, we can transform this construction to a weighted network congestion game, see the full version [12] for details.

Theorem 3. *Let C be a set of strictly increasing, positive and twice continuously differentiable functions. Then, C is consistent for multi-commodity network games with*

at least three players if and only if one of the following cases holds: (*i*) *C contains only affine functions;* (*ii*) *C contains only exponential functions $c(\ell) = a_c\, e^{\phi \ell} + b_c$ where $a_c, b_c, \phi \in \mathbb{R}$ and ϕ is independent of c.*

For multi-commodity weighted network games with two players, we obtain the following.

Theorem 4. *Let C be a set of strictly increasing, positive and twice continuously differentiable functions. Let $\mathcal{G}^2(C)$ be the set of two-player (single or two)-commodity network games such that cost functions are in C. Then, C is consistent w.r.t. $\mathcal{G}^2(C)$ if and only if C contains monotonic functions such that for all $c_1, c_2 \in C$, there are constants $a, b \in \mathbb{R}$ with $c_1 = a\, c_2 + b$.*

Single-commodity Networks. Considering weighted single-commodity network congestion games we conclude with a result concerning the FIP.

Theorem 5. *Let C be a set of strictly increasing, positive and twice continuously differentiable functions. Then, C is FIP-consistent for single-commodity network games with at least three players if and only if one of the following cases holds:* (*i*) *C contains only affine functions;* (*ii*) *C contains only exponential functions $c(\ell) = a_c\, e^{\phi \ell} + b_c$ where $a_c, b_c, \phi \in \mathbb{R}$ and ϕ is independent of c.*

7 Conclusions

We obtained an almost complete characterization of consistency of cost functions in weighted congestion games. The following issues are open. We required that cost functions are twice-continuously differentiable. Although almost all practically relevant functions satisfy this condition, it would be interesting to weaken this assumption.

For single-commodity games with at least three players, we were only able to characterize the FIP, not consistency. The single-commodity case, however, behaves completely different as, for instance, Anshelevich et al. [3] have shown that for positive and strictly decreasing cost functions, there is always a PNE.

Acknowledgment

We thank Hans-Christian Kreusler for contributing to Lemma 1.

References

1. Ackermann, H., Röglin, H., Vöcking, B.: Pure Nash equilibria in player-specific and weighted congestion games. Theor. Comput. Sci. 410(17), 1552–1563 (2009)
2. Aland, S., Dumrauf, D., Gairing, M., Monien, B., Schoppmann, F.: Exact price of anarchy for polynomial congestion games. In: Proc. of STACS, pp. 218–229 (2006)
3. Anshelevich, E., Dasgupta, A., Kleinberg, J., Tardos, E., Wexler, T., Roughgarden, T.: The price of stability for network design with fair cost allocation. In: Proc. of FOCS, pp. 295–304 (2004)

4. Awerbuch, B., Azar, Y., Epstein, A.: The price of routing unsplittable flow. In: Proc. of STOC, pp. 57–66 (2005)
5. Christodoulou, G., Koutsoupias, E.: The price of anarchy of finite congestion games. In: Proc. of STOC, pp. 67–73 (2005)
6. Dunkel, J., Schulz, A.S.: On the complexity of pure-strategy Nash equilibria in congestion and local-effect games. Math. Oper. Res. 33(4), 851–868 (2008)
7. Fanelli, A., Moscardelli, L.: On best response dynamics in weighted congestion games with polynomial delays. In: Leonardi, S. (ed.) WINE 2009. LNCS, vol. 5929, pp. 55–66. Springer, Heidelberg (2009)
8. Fotakis, D., Kontogiannis, S., Spirakis, P.G.: Selfish unsplittable flows. Theor. Comput. Sci. 348(2-3), 226–239 (2005)
9. Fotakis, D., Kontogiannis, S., Spirakis, P.G.: Atomic congestion games among coalitions. In: Proc. of ICALP, pp. 572–583 (2006)
10. Gairing, M., Monien, B., Tiemann, K.: Routing (un-) splittable flow in games with player-specific linear latency functions. In: Proc. of ICALP, pp. 501–512 (2006)
11. Goemans, M., Mirrokni, V., Vetta, A.: Sink equilibria and convergence. In: Proc. of FOCS, pp. 142–154 (2005)
12. Harks, T., Klimm, M.: On the existence of pure Nash equilibria in weighted congestion games. Technical Report 1, COGA, TU Berlin (2010)
13. Harks, T., Klimm, M., Möhring, R.H.: Characterizing the existence of potential functions in weighted congestion games. In: Proc. of SAGT, pp. 97–108 (2009)
14. Ieong, S., McGrew, R., Nudelman, E., Shoham, Y., Sun, Q.: Fast and compact: A simple class of congestion games. In: Proc. of AAAI, pp. 489–494 (2005)
15. Libman, L., Orda, A.: Atomic resource sharing in noncooperative networks. Telecommunication Systems 17(4), 385–409 (2001)
16. Milchtaich, I.: The equilibrium existence problem in finite network congestion games. In: Spirakis, P.G., Mavronicolas, M., Kontogiannis, S.C. (eds.) WINE 2006. LNCS, vol. 4286, pp. 87–98. Springer, Heidelberg (2006)
17. Panagopoulou, P.N., Spirakis, P.G.: Algorithms for pure Nash equilibria in weighted congestion games. ACM Journal of Experimental Algorithmics 11(2.7), 1–19 (2006)
18. Rosenthal, R.W.: A class of games possessing pure-strategy Nash equilibria. International Journal of Game Theory 2(1), 65–67 (1973)

On the Limitations of Greedy Mechanism Design for Truthful Combinatorial Auctions

Allan Borodin and Brendan Lucier

Dept of Computer Science, University of Toronto
{bor,blucier}@cs.toronto.edu

Abstract. We study the combinatorial auction (CA) problem, in which m objects are sold to rational agents and the goal is to maximize social welfare. Of particular interest is the special case in which agents are interested in sets of size at most s (s-CAs), where a simple greedy algorithm obtains an $s+1$ approximation but no truthful algorithm is known to perform better than $O(m/\sqrt{\log m})$. As partial work towards resolving this gap, we ask: what is the power of truthful greedy algorithms for CA problems? The notion of greediness is associated with a broad class of algorithms, known as priority algorithms, which encapsulates many natural auction methods. We show that no truthful greedy priority algorithm can obtain an approximation to the CA problem that is sublinear in m, even for s-CAs with $s \geq 2$.

1 Introduction

The field of algorithmic mechanism design attempts to bridge the competing demands of agent selfishness and computational constraints. The difficulty in such a setting is that agents may lie about their inputs in order to obtain a more desirable outcome. It is often possible to circumvent this obstacle by using payments to elicit truthful responses. Indeed, if the goal of the algorithm is to maximize the total welfare of all agents, the well-known VCG mechanism does precisely that: each agent maximizes his utility by reporting truthfully. However, the VCG mechanism requires that the underlying optimization problem be solved exactly, and is therefore ill-suited for computationally intractable problems. Determining the power of truthful *approximation* mechanisms is a fundamental problem in algorithmic mechanism design.

The combinatorial auction (CA) problem holds a position at the center of this conflict between truthfulness and approximability. Without strategic considerations, one can obtain an $O(\min\{n, \sqrt{m}\})$ approximation for CAs with n bidders and m objects with a conceptually simple (albeit not obvious) greedy algorithm [25], and this is the best possible under standard complexity assumptions [16,31]. However, no deterministic truthful mechanism for multi-minded auctions is known to obtain an approximation ratio better than $O(\frac{m}{\sqrt{\log m}})$ [17]. This is true even for the special case where each bidder is interested only in sets of size at most some constant s (the s-CA problem), where the obvious greedy algorithm obtains an $s+1$ approximation. Whether these gaps are essential for

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 90–101, 2010.

the CA problem, or whether there is some universal process by which approximation algorithms can be made truthful without heavy loss in performance, is a central open question that has received significant attention over the past decade [14,21,25,28,30].

A lower bound for the related combinatorial public project problem [30] shows that there is a large asymptotic gap separating approximation by deterministic algorithms and by deterministic truthful mechanisms in general allocation problems. Currently, the only such lower bounds known for the CA problem are limited to max-in-range (MIR) algorithms [9]. While many known truthful CA algorithms are MIR, the possibility yet remains that non-MIR algorithms could be used to bridge the gap between truthful and non-truthful CA design. We consider lower bounds for truthful CAs by focusing on an alternative class of algorithms. We ask: can any truthful *greedy* algorithm obtain an approximation ratio better than $O(\frac{m}{\sqrt{\log(m)}})$? Our interest in greedy algorithms is motivated threefold. First, most known examples of truthful, non-MIR algorithms for combinatorial auction problems apply greedy methods [1,4,8,10,20,24,25,28]; indeed, greedy algorithms embody the conceptual monotonicity properties generally associated with truthfulness, and are thus natural candidates for truthful mechanism construction. Second, simple greedy auctions are often used in practice, despite the fact that they are not incentive compatible; this leads us to suspect that they are good candidates for auctions due to other considerations, such as ease of public understanding. Finally, greedy algorithms are known to obtain asymptotically tight approximation bounds for many CA problems despite their simplicity.

We use the term "greedy algorithm" to refer to any of a large class of algorithms known as *priority algorithms* [7]. The class of priority algorithms captures a general notion of greedy algorithm behaviour. Priority algorithms include, for example, many well-known primal-dual algorithms, as well as other greedy algorithms with adaptive and non-trivial selection rules. Moreover, this class is *independent of computational constraints* and also independent of the manner in which valuation functions are accessed. In particular, our results apply to algorithms in the demand query model and the general query model, as well as to auctions in which bids are explicitly represented. Roughly speaking, a priority algorithm has some notion of what consistutes the "best" bid in any given auction instance; the auction finds this bid, satisfies it, then iteratively resolves the reduced auction problem with fewer objects (possibly with an adaptive notion of the "best" bid). For example, the truthful algorithm for multi-unit auctions due to Bartal et al. [4] that updates a price vector while iteratively satisfying agent demands falls into this framework. Our main result demonstrates that if a truthful auction for an s-CA proceeds in this way, then it cannot perform much better than the trivial algorithm that allocates all objects to a single bidder.

Theorem: No deterministic truthful priority algorithm (defined formally in the text) for the CA problem obtains an $o(\min\{m, n\})$ approximation to the optimal social welfare (even for s-CAs with $s \geq 2$).

The gap described in our result is extreme: for $s = 2$, the standard (but non-truthful) greedy algorithm is a 3-approximation for the s-CA problem, but no truthful greedy algorithm can obtain a sublinear approximation bound.

We also consider the combinatorial auction problem for submodular bidders (SMCA), which has been the focus of much study [13,12,19,24]. We study a class of greedy algorithms that is especially well-suited to the SMCA problem. Such algorithms consider the objects of the auction one at a time and greedily assign them to bidders to maximize marginal utilities. It was shown in [24] that any such algorithm[1] attains a 2-approximation to the SMCA problem, but that not all are incentive compatible. We show that, in fact, no such algorithm can be incentive compatible.

Theorem: Any deterministic algorithm for submodular combinatorial auctions that considers objects and assigns them in order to maximize marginal utility cannot obtain a bounded approximation to the optimal social welfare.

1.1 Related Work

Many truthful approximation mechanisms are known for CAs with single-minded bidders. Following the Lehmann et al. [25] truthful greedy mechanism for single-minded CAs, Mu'alem and Nisan [28] showed that any *monotone* greedy algorithm for single-minded bidders is truthful, and outlined various techniques for combining approximation algorithms while retaining truthfulness. This led to the development of many other truthful algorithms in single-minded settings [2,8] and additional construction techniques, such as the iterative greedy packing of loser-independent algorithms due to Chekuri and Gamzu [10].

Less is known in the setting of general bidder valuations. Bartal et al. [4] give a greedy algorithm for multi-unit CAs that obtains an $O(Bm^{\frac{1}{B-2}})$ approximation when there are B copies of each object. Lavi and Swamy [23] give a general method for constructing randomized mechanisms that are truthful in expectation, meaning that agents maximize their expected utility by declaring truthfully. Their construction generates a k-approximate mechanism from an LP for which there is an algorithm that verifies a k-integrality gap. In the applications they discuss, these verifiers take the form of greedy algorithms, which play a prominant role in the final mechanisms.

A significant line of research aims to give lower bounds on the approximating power of deterministic truthful algorithms for CAs. Lehmann, Mu'alem, and Nisan [21] show that any truthful CA mechanism that uses a suitable bidding language, is unanimity-respecting, and satisfies the independence of irrelevant alternatives property (IIA) cannot attain a polynomial approximation ratio. It has also been shown that, roughly speaking, no truthful polytime subadditive combinatorial auction mechanism that is *stable*[2] can obtain an approximation

[1] The degree of freedom in this class of algorithms is the order in which the objects are considered.

[2] In a stable mechanism, no player can alter the outcome (i.e. by changing his declaration) without causing his own allocated set to change.

ratio better than 2 [14]. Also, no max-in-range algorithm can obtain an approximation ratio better than $\Omega(\sqrt{m})$ when agents have budget-constrained additive valuations [9]. These lower bounds are incomparable to our own, as priority algorithms need not be MIR, stable, unanimity-respecting, or satisfy IIA[3].

Another line of work gives lower bounds for greedy algorithms without truthfulness restrictions. Gonen and Lehmann [15] showed that no algorithm that greedily accepts bids for sets can guarantee an approximation better than $\sqrt{m}$. Similarly, Krysta [20] showed that no oblivious greedy algorithm (in our terminology: fixed order greedy priority algorithm) obtains approximation ratio better than $\sqrt{m}$. (In fact, Krysta derives this bound for a more general class of problems that includes multi-unit CAs.) In contrast, we consider the more general class of priority algorithms but restrict them to be incentive-compatible.

The class of priority algorithms is loosely related to the notion of online algorithms. Mechanism design has been studied in a number of online settings, and lower bounds are known for the performance of truthful algorithms in these settings [22,27]. The critical difference between these results and our lower bounds is that a priority algorithm has control over the order in which input items are considered, whereas in an online setting this order is chosen adversarily.

In contrast to the negative results of this paper, greedy algorithms can provide good approximations when rational agents are assumed to bid at Bayes-Nash equilibria. In particular, there is a greedy combinatorial auction for submodular agents that obtains a 2-approximation at equilibrium [11], and the greedy GSP auction for internet advertising can be shown to obtain a 1.6-approximation at equilibrium [26]. Recently, we have shown [6] that, in a wide variety of contexts, c-approximate monotone greedy allocations can be made into mechanisms whose Bayes-Nash equilibria yield $c(1 + o(1))$ approximations.

2 Definitions and Preliminary Results

Combinatorial Auctions. A *combinatorial auction* consists of n bidders and a set M of m objects. Each bidder i has a value for each subset of objects $S \subseteq M$, described by a valuation function $v_i : 2^M \to \mathbb{R}$ which we call the *type* of agent i. We assume each v_i is monotone and normalized so that $v_i(\emptyset) = 0$. We denote by V_i the space of all possible valuation functions for agent i, and $V = V_1 \times V_2 \times \cdots \times V_n$. We write $\mathbf{v}$ for a profile of n valuation functions, one per agent, and $\mathbf{v}_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$, so that $\mathbf{v} = (v_i, v_{-i})$.

A valuation function v is *single-minded* if there exists a set $S \subseteq M$ and a value $x \geq 0$ such that, for all $T \subseteq M$, $v(T) = x$ if $S \subseteq T$ and 0 otherwise. A valuation function v is *k-minded* if it is the maximum of k single-minded functions. That is, there exist k sets $S_1, \ldots, S_k$ such that for all subsets $T \subseteq M$ we have $v(T) = \max\{v(S_i) | S_i \subseteq T\}$. An *additive* valuation function v is specified

[3] The notion of IIA has been associated with priority algorithms, but in a different context than in [21]. In mechanism design IIA is a property of the mapping between input valuations and output allocations, whereas for priority algorithms the term IIA describes restrictions on the order in which input items can be considered.

by m values $x_1, \ldots, x_m \in \mathbb{R}_{\geq 0}$ so that $v(T) = \sum_{a_i \in T} x_i$. A valuation function v is *submodular* if it satisfies $v(T) + v(S) \geq v(S \cup T) + v(S \cap T)$ for all $S, T \subseteq M$.

A *direct revelation mechanism* (or just *mechanism*) $\mathcal{M} = (G, P)$ consists of an *allocation algorithm* G and a *payment algorithm* P. Given valuation profile $\mathbf{d}$, $G(\mathbf{d})$ returns an allocation of objects to bidders, and $P(\mathbf{d})$ returns the payment extracted from each agent. For each agent i we write $G_i(\mathbf{d})$ and $P_i(\mathbf{d})$ for the set given to and payment extracted from i. We think of $\mathbf{d}$ as a profile of declared valuations made by the agents to the mechanism. The *social welfare* obtained by G on declaration $\mathbf{d}$ is $SW(\mathbf{d}) = \sum_{i \in N} d_i(G_i(\mathbf{d}))$. The *optimal social welfare*, SW_{opt}, is the maximum of $\sum_{i \in N} t_i(S_i)$ over all valid allocations $(S_1, \ldots, S_n)$. Algorithm G is a c-approximation if $SW(\mathbf{t}) \geq \frac{1}{c} SW_{opt}$ for all type profiles $\mathbf{t}$.

Fixing mechanism $\mathcal{M}$ and type profile $\mathbf{t}$, the *utility* of bidder i given declaration $\mathbf{d}$ is $u_i(\mathbf{d}) = t_i(G_i(\mathbf{d})) - P_i(\mathbf{d})$. Mechanism $\mathcal{M}$ is *truthful* (or *incentive compatible*) if for every type profile $\mathbf{t}$, agent i, and declaration profile $\mathbf{d}$, $u_i(t_i, \mathbf{d}_{-i}) \geq u_i(\mathbf{d})$. That is, agent i maximizes his utility by declaring his type, regardless of the declarations of the other agents. We say that G is truthful if there exists a payment function P such that the mechanism (G, P) is truthful.

Critical Prices. From Bartal, Gonen and Nisan [4], we have the following characterization of truthful CA mechanisms.

Theorem 1. *A mechanism is truthful if and only if, for every i, S, and $\mathbf{d}_{-i}$, there is a price $p_i(S, \mathbf{d}_{-i})$ such that whenever bidder i is allocated S his payment is $p_i(S, \mathbf{d}_{-i})$, and agent i is allocated a set S_i that maximizes $d_i(S_i) - p_i(S_i, \mathbf{d}_{-i})$.*

We refer to $p_i(S, \mathbf{d}_{-i})$ as the *critical price* of S for agent i. Note that $p_i(S, \mathbf{d}_{-i})$ need not be finite: if $p_i(S, \mathbf{d}_{-i}) = \infty$ then the mechanism will not allocate S to bidder i for any reported valuation d_i. In addition, one can assume without loss of generality that critical prices are monotone.

Priority Algorithms. We view an input instance to an algorithm as a selection of *input items* from a known input space $\mathcal{I}$. Note that $\mathcal{I}$ depends on the problem being considered, and is the set of *all possible* input items: an input instance is a finite subset of $\mathcal{I}$. The problem definition may place restrictions on the input: an input instance $I \subseteq \mathcal{I}$ is *valid* if it satisfies all such restrictions. The output of the algorithm is a decision made for each input item. For example, these decisions may be of the form "accept/reject", allocate set S to agent i, etc. The problem may place restrictions on the nature of the decisions made by the algorithm; we say that the output of the algorithm is *valid* if it satisfies all such restrictions. A *priority algorithm* is then any algorithm of the following form:

Adaptive Priority
Input: A set I of items, $I \subseteq \mathcal{I}$
while not empty(I)
  Ordering: Choose, without looking at I, a total ordering $\mathcal{T}$ over $\mathcal{I}$
  next $\leftarrow$ first item in I according to ordering $\mathcal{T}$
  Decision: make an irrevocable decision for item *next*
  remove *next* from I; remove from $\mathcal{I}$ any items preceding *next* in $\mathcal{T}$
end while

We emphasize the importance of the ordering step in this framework: an adaptive priority algorithm is free to choose *any* ordering over the space of possible input items, and can change this ordering adaptively after each input item is considered. Once an item is processed, the algorithm is not permitted to modify its decision. On each iteration a priority algorithm learns what (higher-priority) items are *not* in the input. A special case of (adaptive) priority algorithms are fixed order priority algorithms in which one fixed ordering is chosen before the while loop (i.e. the "ordering" and "while" statements are interchanged). Our inapproximation results for truthful CAs will hold for the more general class of adaptive priority algorithms.

The term "greedy" implies a more opportunistic aspect than is apparent in the definition of priority algorithms and indeed we view priority algorithms as "greedy-like". A *greedy* priority algorithm satisfies an additional property: the choice made for each input item must optimize the objective of the algorithm as though that item were the last item in the input.

3 Truthful Priority Algorithms

We wish to show that no truthful priority algorithm can provide a non-trivial approximation to social welfare. In order to apply the concept of priority algorithms we must define the set $\mathcal{I}$ of possible input items and the nature of decisions to be made. We consider two natural input formulations: sets as items, and bidders as items. We assume that n, the number of bidders, and m, the number of objects, are known to the mechanism and let $k = \min\{m, n\}$.

3.1 Sets as Items

In our primary model, we view an input instance to the combinatorial auction problem as a list of set-value pairs for each bidder. An item is a tuple (i, S, t), $i \in N$, $S \subseteq M$, and $t \in \mathbb{R}_{\geq 0}$. A valid input instance $I \subset \mathcal{I}$ contains at most one tuple $(i, S, v_i(S))$ for each $i \in N$ and $S \subseteq M$ and for every pair of tuples (i, S, v) and (i', S', v') in I such that $i = i'$ and $S \subseteq S'$, it must be that $v \leq v'$. We note that since a valid input instance may contain an exponential number of items, this model applies most directly to algorithms that use oracles to query input valuations, such as demand oracles[4], but it can also apply to succinctly represented valuation functions.[5]

The decision to be made for item (i, S, t) is whether or not the objects in S should be added to any objects already allocated to bidder i. For example, an

[4] It is tempting to assume that this model is equivalent to a value query model, where the mechanism queries bidders for their values for given sets. The priority algorithm model is actually more general, as the mechanism is free to choose an arbitrary ordering over the space of possible set/value combinations. In particular, the mechanism could order the set/value pairs by the utility they would generate under a given set of additive prices, simulating a demand query oracle.

[5] That is, by assigning priority only to those tuples appearing in a given representation.

algorithm may consider item (i, S_1, t_1) and decide to allocate S_1 to bidder i, then later consider another item (i, S_2, t_2) (where S_2 and S_1 are not necessarily disjoint) and, if feasible, decide to change bidder i's allocation to $S_1 \cup S_2$.

A *greedy algorithm* in the sets as items model must accept any feasible, profitable item (i, S, t) it considers[6]. Our main result is a lower bound on the approximation ratio achievable by a truthful greedy algorithm in the sets as items model.

Theorem 2. *Suppose A is an incentive compatible greedy priority algorithm that uses sets as items. Then A cannot approximate the optimal social welfare by a factor of $\frac{(1-\delta)k}{2}$ for any $\delta > 0$. This result also applies to the special case of (3-minded bidders for) the 2-CA problem, in which each desired set has size at most 2.*

Theorem 2 implies a severe separation between the power of greedy algorithms and the power of truthful greedy algorithms. A simple greedy algorithm obtains a 3-approximation for the 2-CA problem, yet no truthful greedy priority algorithm (indeed, any algorithm that irrevocably satisfies bids based on a notion of priority) can obtain even a sublinear approximation.

Proof. Choose $\delta > 0$ and suppose A obtains a bounded approximation ratio. For each $i \in N$, let V_{-i}^+ be the set of valuations with the property that $v_\ell(S) > 0$ for all $\ell \neq i$ and all non-empty $S \subseteq M$. The heart of our proof is the following claim, which shows that the relationship between critical prices for singletons for one bidder is independent of the valuations of other bidders. Recall that $p_i(S, \mathbf{d}_{-i})$ is the critical price for set S for bidder i, given $\mathbf{d}_{-i}$.

Lemma 1. *For all $i \in N$, and for all $a, b \in M$, either $p_i(\{a\}, \mathbf{d}_{-i}) \geq p_i(\{b\}, \mathbf{d}_{-i})$ for all $\mathbf{d}_{-i} \in V_{-i}^+$, or $p_i(\{a\}, \mathbf{d}_{-i}) \leq p_i(\{b\}, \mathbf{d}_{-i})$ for all $\mathbf{d}_{-i} \in V_{-i}^+$. This is true even when agents desire sets of size at most 2.*

We can think of Lemma 1 as defining, for each $i \in N$, an ordering over the elements of M. For each $i \in N$ and $a, b \in M$, write $a \preceq_i b$ to mean $p_i(a, \mathbf{d}_{-i}) \leq p_i(b, \mathbf{d}_{-i})$ for all $\mathbf{d}_{-i} \in V_{-1}^+$. For all $i \in N$ and $a \in M$, define $T_i(a) = \{a_j : a \preceq_i a_j\}$. That is, $T_i(a)$ is the set of objects that have higher price than a for agent i. Our next claim shows a strong relationship between whether a is allocated to bidder i and whether any object in $T_i(a)$ is allocated to bidder i.

Lemma 2. *Choose $a \in M$, $i \in N$, and $S \subseteq M$, and suppose $S \cap T_i(a) \neq \emptyset$. Choose some $v_i \in V_i$ and suppose that $v_i(a) > v_i(S)$. Then if $\mathbf{v}_{-i} \in V_{-i}^+$, bidder i cannot be allocated set S by algorithm A given input $\mathbf{v}$.*

Lemma 2 is strongest when $T_i(a)$ is large; that is, when a is "small" in the ordering $\preceq_i$. We therefore wish to find an object of M that is small according to many of these orderings, simultaneously. Let $R(a) = \{i \in N : |T_i(a)| \geq k/2\}$, so $R(a)$ is the set of players for which there are at least $k/2$ objects greater than a. The next claim follows by a straightforward counting argument.

[6] That is, any item (i, S, t) such that no objects in S have already been allocated to another bidder and $t > 0$.

Lemma 3. *There exists $a^* \in M$ such that $|R(a^*)| \geq k/2$.*

We are now ready to proceed with the proof of Theorem 2. Let $a^* \in M$ be the object from Lemma 3. Let $\epsilon > 0$ be a sufficiently small value to be defined later. We now define a particular input instance to algorithm A. For each $i \in R(a^*)$, bidder i will declare the following valuation function, v_i:

$$v_i(S) = \begin{cases} 1 & \text{if } a^* \in S \\ 1 - \delta/2 & \text{if } a^* \notin S \text{ and } S \cap (T_i(a^*)) \neq \emptyset \\ \epsilon & \text{otherwise.} \end{cases}$$

Each bidder $i \notin R(a^*)$ will declare a value of ϵ for every set.

For each $i \in R(a^*)$, $v_i(a_j) \geq 1-\delta/2$ for every $a_j \in T_i(a^*)$. Since $|R(a^*)| \geq k/2$ and $|T_i(a^*)| \geq k/2$, it is possible to obtain a social welfare of at least $\frac{(1-\delta/2)k}{2}$ by allocating singletons to bidders in $R(a^*)$.

Consider the social welfare obtained by algorithm A. The algorithm can allocate object a^* to at most one bidder, say bidder i, who will obtain a social welfare of at most 1. For any bidder $\ell \in R(a^*)$, $\ell \neq i$, $v_\ell(S) = 1 - \delta/2 < 1$ for any S containing elements of $T_\ell(a^*)$ but not a^*. Thus, by Lemma 2, no bidder in $R(a^*)$ can be allocated any set S that contains an element of $T_i(a^*)$ but not a^*. Therefore every bidder other than bidder i can obtain a value of at most ϵ, for a total social welfare of at most $1 + k\epsilon$.

We conclude that algorithm A has an approximation factor no better than $\frac{k(1-\delta/2)}{2(1+k\epsilon)}$. Choosing $\epsilon < \frac{\delta}{2(1-\delta)k}$ yields an approximation ratio greater than $\frac{k(1-\delta)}{2}$, completing the proof of Theorem 2.

We believe that the greediness assumption of Theorem 2 can be removed, but we leave this as an open problem. As partial progress we show that this is true for the following (more restricted) model of priority algorithms, in which an algorithm can only consider and allocate sets whose values are not implied by the values of other sets.

Elementary bids as items. Consider an auction setting in which agents do not provide entire valuation functions, but rather each agent specifies a list of *desired sets* $S_1, \ldots, S_k$ and a value for each one. Moreover, each agent receives either a desired set or the empty set. This can be thought of as an auction with a succinct representation for valuation functions, in the spirit of the XOR bidding language [29]. We model such an auction as a priority algorithm by considering items to be the bids for desired sets. In such a setting, the specified set-value pairs are called *elementary bids*. We say that the priority model uses *elementary bids as items* when only elementary bids $(i, S, v(S))$ can be considered by the algorithm. For each item $(i, S, v(S))$, the decision to be made is whether or not S will be the final set allocated to agent i; that is, whether or not the elementary bid for S will be "satisfied." In particular, unlike in the sets as items model, we do not permit the algorithm to build up an allocation incrementally by accepting many elementary bids from a single agent.

We now show that the greediness assumption from Theorem 2 can be removed when we consider priority algorithms in the elementary bids as items model.

Theorem 3. *Suppose A is an incentive compatible priority algorithm for the CA problem that uses elementary bids as items. Then A cannot approximate the optimal social welfare by a factor of $(1-\delta)k$ for any $\delta > 0$.*

3.2 Bidders as Items

Roughly speaking, the lower bounds in Theorems 2 and 3 follow from a priority algorithm's inability to determine which of many different mutually-exclusive desires of an agent to consider first when constructing an allocation. One might guess that such difficulties can be overcome by presenting an algorithm with more information about an agent's valuation function at each step. To this end, we consider an alternative model of priority algorithms in which the agents themselves are the items, and the algorithm is given complete access to an agent's declared valuation function each round.

Under this model, $\mathcal{I}$ consists of all pairs (i, v_i), where $i \in N$ and $v_i \in V_i$. A valid input instance contains one item for each bidder. The decision to be made for item (i, v_i) is a set $S \subseteq M$ to assign to bidder i. The truthful greedy CA mechanism for single-minded bidders falls within this model, as does its (non-truthful) generalization to complex bidders [25], the primal-dual algorithm of [8], and the (first) algorithm of [4] for multi-unit CAs. We now establish an inapproximation bound for truthful priority allocations that use bidders as items.

Theorem 4. *Suppose A is an incentive compatible priority algorithm for the (2-minded) CA problem that uses bidders as items. Than A cannot approximate the optimal social welfare by a factor of $\frac{(1-\delta)k}{2}$ for any $\delta > 0$.*

4 Truthful Submodular Priority Auctions

Lehmann, Lehmann, and Nisan [24] proposed a class of greedy algorithms that is well-suited to auctions with submodular bidders; namely, objects are considered in any order and incrementally assigned to greedily maximize marginal utility. They showed that any ordering of the objects leads to a 2-approximation of social welfare, but not every ordering of objects leads to an incentive compatible algorithm. However, this does not preclude the possibility of obtaining truthfulness using some adaptive method of ordering the objects.

We consider a model of priority algorithms which uses the m objects as input items. In this model, an item will be represented by an object x, plus the value $v_i(x|S)$ for all $i \in N$ and $S \subseteq M$ (where $v_i(x|S) := v_i(S \cup \{x\}) - v_i(S)$ is the marginal utility of bidder i for item x, given set S). We note that the online greedy algorithm described above falls into this model. We show that no greedy priority algorithm in this model is incentive compatible.

Theorem 5. *Any greedy priority algorithm for the combinatorial auction problem that uses objects as items is not incentive compatible. This holds even if the bidders are assumed to be submodular.*

5 Future Work

The goal of algorithmic mechanism design is the construction of algorithms in situations where inputs are controlled by selfish agents. We considered this fundamental issue in the context of conceptually simple methods (independent of time bounds) rather than in the context of time constrained algorithms. Our results concerning priority algorithms (as a model for greedy mechanisms) is a natural beginning to a more general study of the power and limitations of conceptually simple mechanisms. Even though the priority framework represents a restricted (albeit natural) algorithmic approach, there are still many unresolved questions even for the most basic mechanism design questions. In particular, we believe that the results of Section 3 can be unified to show that the linear inapproximation bound holds for all priority algorithms (without restrictions). The power of greedy algorithms for unit-demand auctions (s-CAs with $s = 1$) is also not understood; it is not difficult to show that optimality cannot be achieved by priority algorithms, but is it possible to obtain a sublinear approximation bound with greedy methods? Even though an optimal polytime algorithm exists for this case, greedy algorithms for the problem are still of interest, evidenced by the use of greedy algorithms in practice to resolve unit-demand AdWord auctions.

An obvious direction of future work is to widen the scope of a systematic search for truthful approximation algorithms; priority algorithms can be extended in many ways. One might consider priority algorithms with a more esoteric input model, such as a hybrid of the sets as items and bidders as items models. Priority algorithms can be extended to allow revocable acceptances [18] whereby a priority algorithm may "de-allocate" sets or objects that had been previously allocated to make a subsequent allocation feasible. Somewhat related is the priority stack model [5] (as a formalization of local ratio/primal dual algorithms [3]) where items (e.g. bidders or bids) initially accepted are placed in a stack and then the stack is popped to ensure feasibility. This is similar to algorithms that allow a priority allocation algorithm to be followed by some simple "cleanup" stage [20]. Another possibility is to consider allocations that are comprised of taking the best of two (or more) priority algorithms. A special case that has been used in the design of efficient truthful combinatorial auction mechanisms [4,8,28] is to optimize between a priority allocation and the naïve allocation that gives all objects to one bidder. Another obvious extension is to consider randomized priority algorithms, potentially in a Bayesian setting. Finally, one could study more general models for algorithms that implement integrality gaps in LP formulations of packing problems; it would be of particular interest if a deterministic truthful k-approximate mechanism could be constructed from an arbitrary packing LP with integrality gap k, essentially derandomizing the construction of Lavi and Swamy [23].

The results in this paper have thus far been restricted to combinatorial auctions but *the basic question* being asked applies to all mechanism design problems. Namely, when can a conceptually simple approximation to the underlying combinatorial optimization problem be converted into an incentive compatible mechanism that achieves (nearly) the same approximation? For example, one

might consider the power of truthful priority mechanisms for approximating unrelated machines scheduling, or for more general integer packing problems.

Acknowledgements

We thank S. Dobzinski, R. Gonen, and S. Micali for helpful discussions.

References

1. Azar, Y., Gamzu, I., Gutner, S.: Truthful unsplittable flow for large capacity networks. In: Proc. 19th ACM Symp. on Parallel Algorithms and Architectures (2007)
2. Babaioff, M., Blumrosen, L.: Computationally-feasible truthful auctions for convex bundles. In: Proc. 7th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (2004)
3. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. Journal of the ACM 48, 1069–1090 (2001)
4. Bartal, Y., Gonen, R., Nisan, N.: Incentive compatible multi unit combinatorial auctions. In: Proc. 9th Conf. on Theoretical Aspects of Rationality and Knowledge (2003)
5. Borodin, A., Cashman, D., Magen, A.: How well can primal-dual and local-ratio algorithms perform? In: Proc. 32nd Intl. Colloq. on Automata, Languages and Programming (2005)
6. Borodin, A., Lucier, B.: Price of anarchy for greedy auctions. In: Proc. 21th ACM Symp. on Discrete Algorithms (2010)
7. Borodin, A., Nielsen, M.N., Rackoff, C.: (incremental) priority algorithms. In: Proc. 13th ACM Symp. on Discrete Algorithms (2002)
8. Briest, P., Krysta, P., Vöcking, B.: Approximation techniques for utilitarian mechanism design. In: Proc. 36th ACM Symp. on Theory of Computing (2005)
9. Buchfuhrer, D., Dughmi, S., Fu, H., Kleinberg, R., Mossel, E., Papadimitriou, C., Schapira, M., Singer, Y., Umans, C.: Inapproximability for vcg-based combinatorial auctions. In: Proc. 21st ACM Symp. on Discrete Algorithms (2010)
10. Chekuri, C., Gamzu, I.: Truthful mechanisms via greedy iterative packing. In: Proc. 12th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (2009)
11. Christodoulou, G., Kovács, A., Schapira, M.: Bayesian combinatorial auctions. In: Proc. 35th Intl. Colloq. on Automata, Languages and Programming, pp. 820–832 (2008)
12. Dobzinski, S., Nisan, N., Schapira, M.: Approximation algorithms for combinatorial auctions with complement-free bidders. In: Proc. 36th ACM Symp. on Theory of Computing (2005)
13. Dobzinski, S., Schapira, M.: An improved approximation algorithm for combinatorial auctions with submodular bidders. In: Proc. 17th ACM Symp. on Discrete Algorithms (2006)
14. Dobzinski, S., Sundararajan, M.: On characterizations of truthful mechanisms for combinatorial auctions and scheduling. In: Proc. 10th ACM Conf. on Electronic Commerce (2008)

15. Gonen, R., Lehmann, D.: Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In: Proc. 2nd ACM Conf. on Electronic Commerce (2000)
16. Hastad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Electronic Colloquium on Computational Complexity 38 (1997)
17. Holzman, R., Kfir-Dahav, N., Monderer, D., Tennenholtz, M.: Bundling equilibrium in combinatiorial auctions. Games and Economic Behavior 47, 104–123 (2004)
18. Horn, S.: One-pass algorithms with revocable acceptances for job interval selection. University of Toronto MSC thesis (2004)
19. Khot, S., Lipton, R., Markakis, E., Mehta, A.: Inapproximability results for combinatorial auctions with submodular utility functions. In: Deng, X., Ye, Y. (eds.) WINE 2005. LNCS, vol. 3828, pp. 92–101. Springer, Heidelberg (2005)
20. Krysta, P.: Greedy approximation via duality for packing, combinatorial auctions and routing. In: Proc. 30th Intl. Symp. on Mathematical Foundations of Computer Science (2005)
21. Lavi, R., Mu'alem, A., Nisan, N.: Towards a characterization of truthful combinatorial auctions. In: Proc. 44th IEEE Symp. on Foundations of Computer Science (2003)
22. Lavi, R., Nisan, N.: Online ascending auctions for gradually expiring goods. In: Proc. 16th ACM Symp. on Discrete Algorithms (2005)
23. Lavi, R., Swamy, C.: Truthful and near-optimal mechanism design via linear programming. In: Proc. 46th IEEE Symp. on Foundations of Computer Science (2005)
24. Lehmann, B., Lehmann, D., Nisan, N.: Combinatorial auctions with decreasing marginal utilities. In: Proc. 3rd ACM Conf. on Electronic Commerce (2001)
25. Lehmann, D., O'Callaghan, L.I., Shoham, Y.: Truth revelation in approximately efficient combinatorial auctions. In: Proc. 1st ACM Conf. on Electronic Commerce, pp. 96–102. ACM Press, New York (1999)
26. Paes Leme, R., Tardos, E.: Sponsored search equilibria for conservative bidders. In: Fifth Workshop on Ad Auctions (2009)
27. Mahdian, M., Saberi, A.: Multi-unit auctions with unknown supply. In: Proc. 8th ACM Conf. on Electronic Commerce (2006)
28. Mu'alem, A., Nisan, N.: Truthful approximation mechanisms for restricted combinatorial auctions. Games and Economic Behavior 64, 612–631 (2008)
29. Nisan, N.: Bidding and allocation in combinatorial auctions. In: Proc. 2nd ACM Conf. on Electronic Commerce (2000)
30. Papadimitriou, C., Schapira, M., Singer, Y.: On the hardness of being truthful. In: Proc. 49th IEEE Symp. on Foundations of Computer Science (2008)
31. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. In: Proc. 37th ACM Symp. on Theory of Computing (2006)

Mean-Payoff Games and Propositional Proofs*

Albert Atserias[1] and Elitza Maneva[2]

[1] Universitat Politècnica de Catalunya
[2] Universitat de Barcelona

Abstract. We associate a CNF-formula to every instance of the mean-payoff game problem in such a way that if the value of the game is non-negative the formula is satisfiable, and if the value of the game is negative the formula has a polynomial-size refutation in Σ_2-Frege (a.k.a. DNF-resolution). This reduces the problem of solving mean-payoff games to the weak automatizability of Σ_2-Frege, and to the interpolation problem for $\Sigma_{2,2}$-Frege. Since the interpolation problem for Σ_1-Frege (i.e. resolution) is solvable in polynomial time, our result is close to optimal up to the computational complexity of solving mean-payoff games. The proof of the main result requires building low-depth formulas that compute the bits of the sum of a constant number of integers in binary notation, and low-complexity proofs of their relevant arithmetic properties.

1 Introduction

A mean-payoff game is played on a weighted directed graph $G = (V, E)$ with an integer weight $w(e)$ on every arc $e \in E$. Starting at an arbitrary vertex u_0, players 0 and 1 alternate in rounds, each extending the path $u_0, u_1, \ldots, u_n$ built up to that point, by adding one more arc $(u_n, u_{n+1}) \in E$ that leaves the current vertex u_n. The goal of player 0 is to maximize $\nu_0 = \liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(u_{i-1}, u_i)$, while the goal of player 1 is to minimize $\nu_1 = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(u_{i-1}, u_i)$.

These games were studied by Ehrenfeucht and Mycielsky [12] who showed that every such game $\mathcal{G}$ has a value $\nu = \nu_{\mathcal{G}}$ such that player 0 has a *positional* strategy that secures $\nu_0 \geq \nu$, and player 1 has a *positional* strategy that secures $\nu_1 \leq \nu$. Here, a positional strategy is one whose moves depend only on the current vertex and not on the history of the play. We say that the game satisfies *positional determinacy*.

Positional determinacy is a property of interest in complexity theory. On one hand it implies that the problem of deciding if a given game has non-negative value (MPG) belongs to NP ∩ co-NP. This follows from the fact that every positional strategy has a short description, and that given a positional strategy for one player, it is possible to determine the best response strategy for the other in polynomial time. The latter was observed by Zwick and Paterson [27] as an application of Karp's algorithm for finding the minimum cycle mean in a digraph

* First author supported by CICYT TIN2007-68005-C04-03 (LOGFAC-2). Second author supported in part by MICINN Ramon y Cajal and CICYT TIN2007-66523 (FORMALISM). This work was done while visiting CRM, Bellaterra, Barcelona.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 102–113, 2010.

[16]. See [27] also for a direct link with Shapley's simple stochastic games. On the other hand, at the time of writing there is no known polynomial-time algorithm for solving mean-payoff games, not even for a special case called parity games that is of prime importance in applications of automata theory, and the body of literature on the topic keeps growing [15,14,8].

For a problem in NP ∩ co-NP for which a polynomial-time algorithm is not known or obvious, it is compulsory to ask for the nature of the certificates (short proofs of membership) and of the disqualifications (short proofs of non-membership). Celebrated examples where this was insightful are too many to be cited here (see [19,21]). In the case that concerns us, that of mean-payoff games, a new and useful understanding of its membership in NP ∩ co-NP emerges from the combination of two recent results.

The starting point is the observation that the problem MPG reduces to the satisfiability problem for sets of *max-atoms*. A max-atom is an inequality of the form $x_0 \leq \max\{x_1 + a_1, \ldots, x_r + a_r\}$ where $x_0, \ldots, x_r$ are integer variables, and $a_1, \ldots, a_r$ are integer constants. This was first seen in [20] in the special context of scheduling and precedence constraints (with slightly different notation and definitions). The second result is from [7], where the satisfiability problem for max-atoms was re-discovered and given its name, and the problem was studied from the perspective of logic. The authors of [7] introduced an inference system, called *chaining*, that derives new max-atoms that follow from previous ones by simple rules. They showed that this system is both complete and, interestingly, polynomially bounded: if the collection of max-atom inequalities is unsatisfiable, then it has a refutation whose total size is polynomial in the size of the input.

Given these two results, the situation is that for a given mean-payoff game $\mathcal{G}$, a satisfying assignment to the corresponding instance of the max-atom problem is a certificate that $\nu_G \geq 0$, and a refutation of this instance in the chaining inference system is a certificate that $\nu_G < 0$. Therefore MPG reduces to the proof-search problem for this inference system. We address the question whether it also reduces to the proof-search problem for some standard proof-system for propositional logic. In brief, our main result is that a Boolean encoding of the instance expressing $\nu_{\mathcal{G}} \geq 0$ is either satisfiable, or has polynomial-size refutations in Σ_2-Frege, the standard inference system for propositional logic restricted to manipulating DNF-formulas. To be placed in context, in our terminology Σ_1-Frege manipulates clauses and is thus equivalent to propositional resolution.

Related work and consequences. The proof-search problem for a proof system P asks, for a given unsatisfiable Boolean formula A, to find a P-refutation of A. We say that P is automatizable if the proof-search problem for P is solvable in time polynomial in the size of the smallest P-proof of A. The weak automatizability problem for P asks, for a given formula A and an integer r given in unary, to distinguish the case when A is satisfiable from the case when A has a P-refutation of size at most r. It is known that this problem is solvable in polynomial time if and only if there is an automatizable proof system that simulates P.

The question whether some standard proof system is automatizable was introduced in [11], following the work in [18]. These works showed that

extended-Frege and its weaker version TC^0-Frege are not automatizable unless there is a polynomial-time algorithm for factoring. Extended-Frege and TC^0-Frege are the standard inference systems for propositional logic restricted to manipulating Boolean circuits and threshold formulas of bounded depth, respectively. Indeed, their result is stronger since in both cases it shows that there is a reduction from factoring to the weak automatizability problem. To date, the weakest proof system that seems not weakly automatizable is AC^0-Frege, the standard system restricted to Boolean formulas of bounded alternation-depth. But here the hardness result is much weaker since the reduction from factoring is only subexponential and degrades with the target depth of the AC^0-formulas [10].

All these hardness results proceed by exhibiting short refutations of an unsatisfiable Boolean formula that comes from a cryptography-inspired problem based on the hardness of factoring. Since the usual cryptographic primitives require either complex computations or complex proofs of correctness, going below polynomial-size TC^0-Frege or subexponential-size AC^0-Frege is difficult. In particular, there is no clear evidence in favour or against whether Σ_d-Frege, for fixed $d \geq 1$, is weakly automatizable, where Σ_d-formulas are AC^0-formulas of alternation-depth $d-1$ and a disjunction at the root. Not even for Σ_1-Frege (i.e. resolution) there is clear consensus in favour or against it, despite the partial positive results in [6,5] and the partial negative results in [1].

The first consequence of our result is that the problem of solving mean-payoff games reduces to the weak-automatizability of Σ_2-Frege. Our initial goal was to reduce it to the weak-automatizability of resolution, or cutting planes, but these remain open. Note that cutting planes is a natural candidate in the context of max-atoms as it works with linear inequalities over the integers. The difficulty seems to be in simulating disjunctions of inequalities.

A second consequence of our result concerns the problem of interpolation for a proof system P. This is the problem that asks, for a given P-refutation of an unsatisfiable formula of the form $A_0(x, y_0) \wedge A_1(x, y_1)$ and a given truth assignment a for x, to return an $i \in \{0, 1\}$ such that $A_i(a, y_i)$ is itself unsatisfiable. If the feasible interpolation problem for P is solvable in polynomial time we say that P enjoys feasible interpolation. It is known that feasible interpolation is closely related to weak automatizability in the sense that if a system is weakly automatizable, then it enjoys feasible interpolation [11,25]. Proof systems enjoying feasible interpolation include resolution [17], cutting planes [23,9], Lovász-Schrijver [24], and Hilbert's nullstellensatz [26]. On the negative side, it turns out that all known negative results for weak automatizability mentioned above were shown by reducing factoring to the interpolation problem. Thus, extended-Frege, TC^0-Frege and AC^0-Frege probably do not enjoy feasible interpolation. For Σ_d-Frege for fixed $d \geq 2$ there is no evidence in favour or against.

In this front our result implies that the problem of solving mean-payoff games reduces to the interpolation problem for $\Sigma_{2,2}$-Frege, where $\Sigma_{2,2}$-formulas are Σ_3-formulas of bottom fan-in two. Note that Σ_1-Frege does enjoy feasible interpolation since it is equivalent to resolution. Thus our result is close to optimal up to the computational complexity of solving mean-payoff games.

Overview of the proof. Given a mean-payoff game $\mathcal{G}$, we want to find an efficient translation of its associated instance of the max-atom problem into a collection of Boolean clauses. Once this is done, and assuming $\nu_G < 0$, we provide a polynomial-size Σ_2-Frege refutation that simulates the polynomial-size chaining-refutation guaranteed to exist by the results in [7].

Executing this plan requires technical work and is the main contribution of this paper. As part of its solution we need efficient depth-two formulas that compute the bits of the sum of a constant number of non-negative integers represented in binary. This was long known for two summands but the extension to more than two summands is not obvious and appears to be new. This turned out to be specially delicate because we need formulas explicit enough to allow polynomial-size depth-two Frege proofs of their basic properties. For example:

$$\frac{x \leq y + a \qquad y \leq z + b}{x \leq z + a + b}.$$

We hope these will be useful in independent contexts. One key fact in our argument is that we use the above with *constants* a and b, which makes the bottom formula equivalent to $x \leq z + (a + b)$. The point is that if a and b were not constants, the number of summands would grow unbounded, and such sums are known to be not definable by polynomial-size formulas of constant depth [13].

Structure of the paper. In Section 2 we discuss the transformation from mean-payoff games to the max-atom problem, and the chaining inference system. In Section 3 we introduce the notation about Boolean formulas and the definition of Σ_d-Frege. In Section 4 we define the formula $\mathrm{CARRY}(x_1, \ldots, x_r)$ that computes the carry-bit of the sum of r integers given in binary. In Section 5 we simulate the rules of chaining using formal proofs for the arithmetic properties of CARRY. In Section 6 we put everything together and get consequences for proof complexity.

2 Max-atom Refutations

From mean-payoff games to max-atom inequalities. Let $\mathcal{G} = (V, E, V_0, V_1, w)$ be a mean-payoff game, which means that (V, E) is a directed graph with out-degree at least one, $V = V_0 \cup V_1$ is a partition of the vertices, and $w : E \to \{-W, \ldots, 0, \ldots, W\}$ is an integer weight-assignment to the arcs of the graph. This specifies an instance of the mean-payoff game problem which asks whether $\nu \geq 0$. Here, $\nu = \min_{u \in V} \nu(u)$ and $\nu(u)$ is the value of the game started at u. This is defined as $\nu(u) = \sup_{s_0} \inf_{s_1} \nu(u, s_0, s_1)$, where s_0 and s_1 are strategies for player 0 and player 1, and $\nu(u, s_0, s_1) = \liminf_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} w(u_{i-1}, u_i)$ where $u_0 = u$ and $u_{i+1} = s_j(u_0, \ldots, u_i)$ if $u_i \in V_j$ for $j \in \{0, 1\}$.

To every mean-payoff game $\mathcal{G}$ we associate a collection of max-atom inequalities $I(\mathcal{G})$ that is satisfiable if and only if $\nu \geq 0$. This was done for the first time in [20, Lemma 7.5]. Here we give a similar construction discussed in [4].

For every $u \in V$, we introduce one integer variable x_u. For $u \in V_0$, we add $x_u \leq \max\{x_v + w(u, v) : v \in N(u)\}$, where $N(u)$ is the set of out-neighbors of u in G. For $u \in V_1$, we want to impose that $x_u \leq \min\{x_v + w(u, v) : v \in N(u)\}$.

If $N(u) = \{v_1, \ldots, v_h\}$ this is simply $x_u \leq \max\{x_{v_i} + w(u, v_i)\}$ as i ranges over $[h]$. Note that $I(\mathcal{G})$ consists of at most $|E|$ max-atoms involving $|V|$ variables and integer constants in the range $[-W, W]$. Its size is thus polynomial in the size of $\mathcal{G}$. At this point we transformed the question whether $\nu \geq 0$ to the satisfiability of a system of max-atom inequalities. The correctness of the transformation is stated in Lemma 1 below.

Chaining refutations. An offset is a term of the form $x + c$, where x is an integer variable and c is an integer constant. In the following, the letters R and S refer to collections of offsets. Also, if a is an integer constant, $S + a$ refers to the collection of offsets of the form $x + (c + a)$ as $x + c$ ranges over all offsets in S. The inference system introduced in [7] called *chaining* works with max-atom inequalities and has three rules: chaining, simplification and contraction.

1. $x \leq \max(R, y + a)$ and $y \leq \max(S)$ yield $x \leq \max(R, T)$, if $T = S + a$,
2. $x \leq \max(R, x + a)$ yields $x \leq \max(R)$, if $a < 0$,
3. $x \leq \max(R, y + a, y + b)$ yields $x \leq \max(R, y + c)$, if $a \leq c$ and $b \leq c$.

A chaining refutation is a proof of $x \leq \max()$, which is clearly unsatisfiable.

This inference system is sound and complete for refuting unsatisfiable collections of max-atom inequalities [7, Theorem 2]. Even more, it is *polynomially bounded*, which means that if $\mathcal{I}$ is an unsatisfiable collection of max-atoms, then there is a chaining refutation of length polynomial in the size of $\mathcal{I}$, and with numbers of bit-length polynomial in the size of $\mathcal{I}$. This follows from two facts: that if $\mathcal{I}$ is unsatisfiable then it contains an unsatisfiable subcollection where every variable appears at most once on the left-hand side (Lemma 5 in [7]), and that for such subcollections the refutation produced by the completeness proof is polynomial (see the proof of Theorem 4 in [7]).

Putting it all together we get:

Lemma 1 ([20,7,4]). *Let $\mathcal{G}$ be a mean-payoff game with n vertices, m edges, and weights in $[-W, W]$. Let $\mathcal{I} = I(\mathcal{G})$. The following are equivalent:*

1. *$\nu_{\mathcal{G}} < 0$,*
2. *$\mathcal{I}$ is unsatisfiable,*
3. *$\mathcal{I}$ is not satisfied by any assignment with values in $[0, Wm]$,*
4. *$\mathcal{I}$ has a chaining refutation,*
5. *$\mathcal{I}$ has a chaining refutation of length n^2 with constants in $[-Wm, Wm]$.*

3 Preliminaries in Propositional Logic

Boolean formulas. Let $x_1, x_2, \ldots$ be Boolean variables. A literal is either a variable x_i, or its negation $\overline{x_i}$, or the constant 0 or 1. We use literals to build Boolean formulas with the usual connectives. We think of conjunctions and disjunctions as symmetric connectives of unbounded arity.

If A is a set of formulas, we write $\wedge A$ for the conjunction of all formulas in A. Similarly, for $\vee A$. We think of $\neg \wedge A$ and $\neg \vee A$ as the same formulas as $\vee \neg A$ and

$\wedge\neg A$, where $\neg A$ denotes the set of negations of formulas in A. If F is a literal, its size $s(F)$ is 1. If F is a conjunction $\wedge A$ or a disjunction $\vee A$, its size $s(F)$ is $1 + \sum_{G\in A} s(G)$. If $F(1), \ldots, F(r)$ are formulas, we use the notation

$$(\forall i : 1 \leq i \leq r)(F(i)) \equiv F(1) \wedge \cdots \wedge F(r),$$
$$(\exists i : 1 \leq i \leq r)(F(i)) \equiv F(1) \vee \cdots \vee F(r).$$

A clause is a disjunction of literals and a term is a conjunction of literals. A CNF-formula is a conjunction of clauses and a DNF-formula is a disjunction of terms. We define a hierarchy of formulas: let $\Sigma_0 = \Pi_0$ be the set of all literals, and for $d \geq 1$, let Σ_d be the set of all formulas of the form $\vee A$, where A is a set of Π_{d-1}-formulas, and let Π_d-formula be the set of all formulas of the form $\wedge A$, where A is a set of Σ_{d-1}-formulas. We write $\Sigma_{d,k}$ and $\Pi_{d,k}$ for the set of all Σ_{d+1}- and Π_{d+1}-formulas with bottom fan-in at most k. For example, $\Sigma_{1,k}$-formulas are k-DNF-formulas, i.e. composed of terms with at most k literals. We use the notation $\Sigma_{d,\mathtt{c}}$ to denote $\Sigma_{d,k}$ for some unspecified constant $k \geq 1$.

Propositional proofs. We define four rules of inference. The four rules are axiom (AXM), weakening (WKG), introduction of conjunction (IOC), and cut (CUT):

$$\frac{}{F \vee \neg F} \qquad \frac{\Delta}{\Delta \vee G} \qquad \frac{\Delta \vee F \qquad \Delta' \vee G}{\Delta \vee \Delta' \vee (F \wedge G)} \qquad \frac{\Delta \vee F \qquad \Delta' \vee \neg F}{\Delta \vee \Delta'},$$

where F and G denote formulas, and Δ and Δ' denote either formulas or the special empty formula $\Box$. The CUT-rule is also known as the resolution rule.

Let $F_1, \ldots, F_r$ and G be formulas. The *assertion* $F_1, \ldots, F_r$ yield G is denoted by $F_1, \ldots, F_r \vdash G$. A proof of this assertion is a finite sequence of formulas $H_1, H_2, \ldots, H_m$ such that $H_m = G$ and for every $i \in [m]$, either $H_i = F_j$ for some $j \in [r]$, or H_i is the conclusion of an inference rule with hypothesis H_j and H_k for some j and k such that $1 \leq j \leq k \leq i-1$. The length of the proof is m. The size of the proof is the sum of the sizes of all involved formulas. A refutation of $F_1, \ldots, F_r$ is a proof of the assertion $F_1, \ldots, F_r \vdash \Box$. If $\mathcal{C}$ is a collection of formulas, a $\mathcal{C}$-Frege proof is one where all formulas belong to $\mathcal{C}$.

The expression "the assertion $F_1, \ldots, F_r \vdash G$ has a polynomial-size $\mathcal{C}$-Frege proof" means that there exists some universal but unspecified polynomial $p(n)$ such that $F_1, \ldots, F_r \vdash G$ has a $\mathcal{C}$-Frege proof of size at most $p(s(F_1) + \cdots + s(F_r) + s(G))$. Similarly, we use $\mathtt{poly}(n)$ to denote a universal but unspecified polynomial function of n, and $\mathtt{c}$ to denote a universal but unspecified constant.

A resolution proof is one where all formulas are clauses and the only allowed rule is CUT. Note that if the only allowed formulas are clauses then IOC is automatically forbidden. Also it is not hard to see that if there is a Σ_1-Frege refutation of $F_1, \ldots, F_r$ of length m, then there is a resolution refutation of $F_1, \ldots, F_r$ of length at most m as well. Therefore resolution and Σ_1-Frege are essentially the same thing. Let us mention that $\Sigma_{1,k}$-Frege is also known as $\mathrm{Res}(k)$, or as k-DNF-resolution. Along these lines, Σ_2-Frege could be called DNF-resolution.

4 Bitwise Linear Arithmetic

The basic $\Sigma_{2,c}$-formula with which we work expresses an inequality. More specifically, it asserts that an addition results in "overflow", or equivalently that there is a carry-bit generated at the left-most position. As a simple example, suppose we want to express that the sum of two B-bit numbers $x = x_1 \dots x_B$ and $y = y_1 \dots y_B$ is at least 2^B. It is not hard to see that the following formula is equivalent to the desired inequality:

$$(\exists p : 1 \leq p \leq B)(x_p = 1 \wedge y_p = 1 \wedge (\forall q : 1 \leq q \leq p-1)(x_q + y_q = 1)).$$

By writing $x_q + y_q = 1$ as a CNF, this is a $\Sigma_{2,2}$-formula. In this section we generalize this formula to an arbitrary number of B-bit numbers.

Let r, k, ℓ and B be positive integers such that $r \leq k \leq 2^\ell - 1 < 2^B$. Let $\mathbf{x} = (x_1, \dots, x_r)$, where each x_i is a string $x_{i,1} \dots x_{i,B}$ of B Boolean variables. We think of $\mathbf{x}$ as a matrix with r rows and B columns. For each column $p \in \{1, \dots, B\}$, let $\mathbf{x}_p = x_{1,p} + \cdots + x_{r,p}$. We interpret $\mathbf{x}_p$ as a symbol in the alphabet $\{0, \dots, r\} \subseteq \{0, \dots, k\}$, and thus the sum of $\mathbf{x}$ as a word in $\{0, \dots, k\}^B$.

We describe an automaton M that decides whether there is overflow in the addition of r B-bit numbers. It is defined to work on the alphabet $\{0, 1, \dots, k\}$, i.e. its input is $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_B$. In general, M has $k+1$ states each indicating a range for the value of the number read so far, which at step p we will denote by $\mathbf{x}_{[p]} = \mathbf{x}_1 2^{p-1} + \mathbf{x}_2 2^{p-2} + \cdots + \mathbf{x}_{p-1} 2^1 + \mathbf{x}_p 2^0$. The $k+1$ states correspond to the ranges $[0, 2^p - k]$, the following $k-1$ single integer intervals $2^p - (k-1), \dots, 2^p - 1$, and $[2^p, k(2^p - 1)]$. We denote these states by $-k$, $-(k-1), \dots, -1$, and 0, respectively. The two extreme states are absorbing, and correspond respectively to the absence and presence of overflow: if $\mathbf{x}_{[p]} \geq 2^p$ then $\mathbf{x} \geq 2^p 2^{B-p} = 2^B$, hence there is overflow; on the other hand, if $\mathbf{x}_{[p]} \leq 2^p - k$ then $\mathbf{x} \leq (2^p - k)2^{B-p} + k(2^{B-p} - 1) = 2^B - 1$, and there is no overflow. The starting state is -1, because $\mathbf{x}_{[0]} = 0 = 2^0 - 1$. The state machine for $k = 5$ is given in Figure 1.

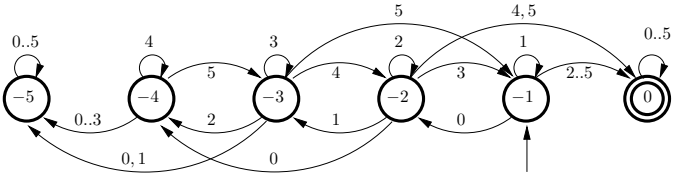


Fig. 1. A state machine deciding if there is overflow in the addition of 5 Boolean strings

The key fact that allows us to design the propositional formula is that if at some stage the machine has not yet reached one of the absorbing states, then we can identify in which intermediate state it is only based on the last ℓ values read, because it suffices to know $\mathbf{x}_{[p]}$ modulo $2^\ell > k - 1$.

We define notation for the number in the last ℓ positions, and the state:

- $A_\ell(\mathbf{x}; p) = \mathbf{x}_1 2^{p-1} + \mathbf{x}_2 2^{p-2} + \cdots + \mathbf{x}_{p-1} 2^1 + \mathbf{x}_p 2^0$ if $0 \le p \le \ell - 1$,
- $A_\ell(\mathbf{x}; p) = \mathbf{x}_{p-\ell+1} 2^{\ell-1} + \mathbf{x}_{p-\ell+2} 2^{\ell-2} + \cdots + \mathbf{x}_{p-1} 2^1 + \mathbf{x}_p 2^0$ if $\ell \le p \le B$,
- $S_\ell(\mathbf{x}; p) = (A_\ell(\mathbf{x}; p) \bmod 2^p) - 2^p$ if $0 \le p \le \ell - 1$,
- $S_\ell(\mathbf{x}; p) = (A_\ell(\mathbf{x}; p) \bmod 2^\ell) - 2^\ell$ if $\ell \le p \le B$,
- $N_\ell(\mathbf{x}; p) = 2S_\ell(\mathbf{x}; p-1) + \mathbf{x}_p$ if $1 \le p \le B$.

Intuitively, $S_\ell(\mathbf{x}; p)$ is the state of the computation of M at time p as long as it did not reach an absorbing state before, and $N_\ell(\mathbf{x}; p)$ stands for "next state".

For every $p \in \{1, \ldots, B\}$, we define the predicates

$$F^+(\mathbf{x}; p) \equiv F^+_{k,\ell}(\mathbf{x}; p) \equiv N_\ell(\mathbf{x}; p) \ge 0,$$
$$F^-(\mathbf{x}; p) \equiv F^-_{k,\ell}(\mathbf{x}; p) \equiv -k < N_\ell(\mathbf{x}; p) < 0.$$

When the parameters k and ℓ are clear from the context we use the lighter notation on the left. Assuming that $S_\ell(\mathbf{x}; p-1)$ is the correct state of M at time $p-1$, the predicate $F^+(\mathbf{x}; p)$ asserts that at time p the automaton accepts, and $F^-(\mathbf{x}; p)$ asserts that at time p the automaton is not at an absorbing state.

Since $F^+(\mathbf{x}; p)$ and $F^-(\mathbf{x}; p)$ depend on no more than $k\ell$ variables of $\mathbf{x}$, those appearing in the definitions of $\mathbf{x}_{p-\ell+1}, \ldots, \mathbf{x}_p$, both $F^+(\mathbf{x}; p)$ and $F^-(\mathbf{x}; p)$ are expressible as $\Sigma_{1,k\ell}$-formulas and as $\Pi_{1,k\ell}$-formulas of size at most $k\ell \cdot 2^{k\ell}$. Using these we define the following formula:

$$\mathrm{CARRY}_{k,\ell}(\mathbf{x}) \equiv (\exists p : 1 \le p \le B)(F^+(\mathbf{x}; p) \wedge (\forall q : 1 \le q \le p-1)(F^-(\mathbf{x}; q)))$$

Intuitively, this formula reads "M eventually accepts". Note that this is a $\Sigma_{2,k\ell}$-formula of size proportional to $B^2 \cdot k\ell \cdot 2^{k\ell}$.

From now on we think of k and ℓ as small and bounded by some universal constant, and of B as unbounded. For concreteness, the uncomfortable reader should fix $k = 11$ and $\ell = 4$ as we will do in later applications.

The letters a, b and c denote B-bit strings $a_1 \ldots a_B$, $b_1 \ldots b_B$ and $c_1 \ldots c_B$, respectively. Abusing notation, we identify a with the number in $[0, 2^B)$ that it represents in binary. This includes the constant $1 = 0^{B-1}1$. The letter z denotes a string of B Boolean variables $z_1 \ldots z_B$. We write $\overline{z}$ for $\overline{z_1} \ldots \overline{z_B}$. The letters $\mathbf{x}$ and $\mathbf{y}$ denote non-empty sequences $(x_1, \ldots, x_{r_x})$ and $(y_1, \ldots, y_{r_y})$, where each x_i is a string of B Boolean variables $x_{i,1} \ldots x_{i,B}$ and each y_i is a string of B Boolean variables $y_{i,1} \ldots y_{i,B}$. Moreover $r_x + r_y + 1 \le k$ and $r_x + 3 \le k$.

Lemma 2. *The following assertions have polynomial-size $\Sigma_{2,\mathrm{c}}$-Frege proofs:*

1. $\mathrm{CARRY}_{k,\ell}(\mathbf{x}, z, 1)$ *and* $\mathrm{CARRY}_{k,\ell}(\mathbf{y}, \overline{z}, 1)$ *yield* $\mathrm{CARRY}_{k,\ell}(\mathbf{x}, \mathbf{y}, 1)$,
2. $\mathrm{CARRY}_{k,\ell}(\mathbf{x}, a, b)$ *yields* $\mathrm{CARRY}_{k,\ell}(\mathbf{x}, c)$, *if* $c = a + b$,
3. $\mathrm{CARRY}_{k,\ell}(\mathbf{x}, a)$ *yields* $\mathrm{CARRY}_{k,\ell}(\mathbf{x}, b)$, *if* $a \le b$,
4. $\mathrm{CARRY}_{k,\ell}(z, \overline{z})$ *yields* $\Box$.

Proof (hint). We sketch *1*, the key proof of the paper. Let M_0, M_1 and M_2 denote the automata on inputs $(\mathbf{x}, z, 1)$, $(\mathbf{y}, \overline{z}, 1)$ and $(\mathbf{x}, \mathbf{y}, 1)$. Let s_o, s_1 and s_2

be their states. We give $\Sigma_{2,\mathtt{c}}$-proofs that while M_0 and M_1 have not yet accepted, $s_2 = s_0 + s_1 + 1$, and if only M_i has accepted, $s_2 \geq s_{1-i} + 1$. These invariants guarantee that whenever both M_0 and M_1 accept, M_2 accepts as well since its state is always *ahead*. We build these proofs by induction on the time-step. The key for staying in $\Sigma_{2,\mathtt{c}}$ is that the invariants are constant-size formulas.

5 Simulating Chaining Refutations

In this section we use the CARRY formula with parameters $k = 11$, $\ell = 4$ and $B = M+2$, where M is a large integer, that we think of as unbounded. As k and ℓ stay fixed everywhere in the section, for convenience we write CARRY instead of $\mathrm{CARRY}_{11,4}$. Note that CARRY is a $\Sigma_{2,44}$-formula.

The letters x, y and z denote integer variables ranging over $[0, 2^M)$, and X, Y and Z denote strings of M Boolean variables for the binary representations of x, y and z. The letters a, b and c denote integer constants in the range $(-2^M, 2^M)$, and A, B and C denote bit-strings of length M for the binary representations of their absolute values $|a|$, $|b|$ and $|c|$. For an integer d in $[0, 2^M)$, we use the notation $\overline{d} = \overline{d}_M$ to denote the integer $2^M - 1 - d$. Note that the M-bit representation of $\overline{d}$ is the bit-wise complement of the M-bit representation of d.

Representing max-atoms. An atom is an expression of the form $x \leq y + a$. We define its $\Sigma_{2,\mathtt{c}}$-representation $\mathrm{R}(x, y, a)$ according to cases $a \geq 0$ and $a < 0$.

Since $x+\overline{x} = 2^M-1$, an atom $x \leq y+a$ with $a \geq 0$ is equivalent to $2^M \leq \overline{x}+y+a+1$, or adding $3\cdot 2^M$ to both sides, to $2^{M+2} \leq \overline{x}+y+a+3\cdot 2^M+1$. As a Boolean formula, we write this as $\mathrm{R}(x, y, a) \equiv \mathrm{CARRY}(00\overline{X}, 00Y, 00A, 110^{M-1}1)$. Note how we padded the strings so that each has length $M + 2$. This padding is necessary to represent arithmetic with negative numbers. Similarly for atoms with $a < 0$ we define $\mathrm{R}(x, y, a) \equiv \mathrm{CARRY}(00\overline{X}, 00Y, 00\overline{A}, 10^{M-1}10)$.

For technical reasons in the proofs we need to view expressions of the form $x \leq y+a+b$ as different from $x \leq y+c$ where $c = a+b$. We represent these as well distinguishing by cases. For example, if $a < 0$ and $b < 0$, the $\Sigma_{2,\mathtt{c}}$-representation is $\mathrm{R}(x, y, a, b) \equiv \mathrm{CARRY}(00\overline{X}, 00Y, 00A, 00\overline{B}, 010^{M-2}11)$.

Let I be the max-atom $x \leq \max\{x_1 + a_1, \ldots, x_r + a_r\}$, where all constants are in the range $(-2^M, 2^M)$. We represent I by the formula

$$(\exists i : 1 \leq i \leq r)(\mathrm{R}(x, x_i, a_i)).$$

Note that this is again a $\Sigma_{2,44}$-formula. We write $F_M(I)$ for this formula and we extend it to sets of max-atoms $\mathcal{I}$ in the obvious way.

Simulating the chaining inference rules. Just as we represent max-atoms by $\Sigma_{2,\mathtt{c}}$-formulas, we can represent the chaining rules by assertions and state:

Theorem 1. *The assertions representing the chaining, simplification and contraction inference rules have polynomial-size $\Sigma_{2,\mathtt{c}}$-Frege proofs.*

Proof (sketch). We use the next lemma which is proved using Lemma 2.

Lemma 3. *The following assertions have polynomial-size $\Sigma_{2,\mathsf{c}}$-Frege proofs:*

1. $\mathrm{R}(x,z,a)$ *and* $\mathrm{R}(z,y,b)$ *yield* $\mathrm{R}(x,y,a,b)$,
2. $\mathrm{R}(x,y,a,b)$ *yields* $\mathrm{R}(x,y,c)$, *if* $c = a+b$,
3. $\mathrm{R}(x,y,a)$ *yields* $\mathrm{R}(x,y,b)$, *if* $a \le b$.

Using this we sketch the proof of the chaining rule. If for every atom A in $x \le \max(R, y+a)$ and every atom B in $y \le \max(S)$ we can prove $A, B \vdash x \le \max(R, S+a)$, the rest will follow from standard manipulation. The only interesting case is when A is $x \le y + a$. Let B be $y \le z_j + b_j$. Part *1* of the lemma gives $x \le z_j + b_j + a$, part *2* gives $x \le z_j + (b_j + a)$, and weakening gives the conclusion. The proof of the simplification rule is similar using case analysis, part *3*, and also Lemma 2.4. For the contraction rule use part *3* again.

6 Main Result and Consequences

Converting to 3-CNF. For a Boolean formula F, let $T = T_3(F)$ denote the standard translation of F into an equi-satisfiable 3-CNF-formula. If $s(F) = s$, then the number of additional variables in T is at most $2s$, and the number of clauses in T is at most $4s$. Also, if F is a $\Sigma_{d,k}$-formula, then the assertion $T \vdash F$ has a polynomial-size $\Sigma_{d,k}$-Frege proof.

Effectively simulating bottom fan-in. Next we discuss the relationship between $\Sigma_{d,k}$-Frege and Σ_d-Frege. The trick was used in [2] for $d = 1$ and was called *effective simulation* in [22].

In general, it is not true that Σ_d-Frege polynomially simulates $\Sigma_{d,k}$-Frege. For example, it is known that Σ_1-Frege does not polynomially simulate $\Sigma_{1,2}$-Frege [3]. However, it *effectively simulates* it. The idea is that if $\mathcal{C}$ is a set of clauses on the variables $x_1, \ldots, x_n$, we can add additional variables z_T and z_C for every possible term T and clause C of at most k literals on the variables $x_1, \ldots, x_n$, and axioms that fix the truth value of the new variables accordingly:

$$\begin{array}{ll} (1)\ z_{C_1 \vee C_2} \leftrightarrow z_{C_1} \vee z_{C_2} & (3)\ z_{x_i} \leftrightarrow x_i \\ (2)\ z_{T_1 \wedge T_2} \leftrightarrow z_{T_1} \wedge z_{T_2} & (4)\ z_{x_i} \leftrightarrow x_i \end{array}$$

Let $E_k(\mathcal{C})$ be the extension of $\mathcal{C}$ with these axioms converted to clauses. Note that if $\mathcal{C}$ is satisfiable, then $E_k(\mathcal{C})$ stays satisfiable: set z_C and z_T to the truth-value of C and T under the truth-assignment satisfying $\mathcal{C}$. On the other hand, if $\mathcal{C}$ is unsatisfiable, the size of the smallest refutation of $\mathcal{C}$ in $\Sigma_{d,k}$-Frege is polynomially related to the size of the smallest refutation of $E_k(\mathcal{C})$ in Σ_d-Frege.

Main result. In the statement of the theorem, the unspecified universal constant in E_{c} is the one from Theorem 1. The proof is a direct consequence of Lemma 1, Theorem 1, and the tricks above.

Theorem 2. *Let $\mathcal{G}$ be a mean-payoff game with m edges and weights in $[-W, W]$. Let M be an integer such that $2^M > Wm$. Let $\mathcal{C} = E_{\mathsf{c}}(T_3(F_M(I(\mathcal{G}))))$. Then:*

1. *if* $\nu_{\mathcal{G}} \geq 0$*, then* $\mathcal{C}$ *is satisfiable,*
2. *if* $\nu_{\mathcal{G}} < 0$*, then* $\mathcal{C}$ *has a polynomial-size* Σ_2*-Frege refutation.*

It is perhaps worth noting that the refutation in this theorem is actually a $\mathrm{Res}(B)$-refutation, where $B = M + 2$ and $M = \lceil \log_2(mW) \rceil + 1$. The reason is that each max-atom is a disjunction of CARRY-formulas with parameter B, and each CARRY-formula with parameter B is a disjunction of conjunctions of fan-in B, with constant fan-in disjunctions at the bottom that end-up wiped away by the E_{c}-trick. Since the size of $\mathcal{C}$ is polynomial in $m \log_2(W)$, this is slightly better than a plain polynomial-size Σ_2-refutation as stated in the theorem.

Consequences for automatizability and interpolation. One direct consequence of Theorem 2 is that if Σ_2-Frege were weakly automatizable, there would be a polynomial-time algorithm for solving mean-payoff games. Indeed, the statement itself of Theorem 2 is a polynomial-time reduction from MPG to the weak automatizability problem for Σ_2-Frege.

On the other hand, there is a tight connection between weak automatizability, interpolation, and the provability of the *reflection principle* (see [25]). We discuss this briefly. Let $\mathrm{SAT}_{n,m}(x, y)$ be a CNF-formula saying that y is an assignment satisfying the CNF-formula encoded by x. Here n and m are the number of variables and the number of clauses of the formula encoded by x. Let $\mathrm{REF}_{n,m,r,d}(x, z)$ be a CNF-formula saying that z is the encoding of a Σ_d-refutation of the CNF-formula encoded by x. Here r is the size of the proof encoded by z. Formalizing this requires some standard encoding of formulas and proofs. Obviously, the formula $\mathrm{SAT}_{n,m}(x, y) \wedge \mathrm{REF}_{n,m,r,d}(x, z)$ is unsatisfiable. This is called the reflection principle for Σ_d-Frege, which happens to have polynomial-size refutations in $\Sigma_{d,2}$-Frege. This was observed in [2] for $d = 1$ and the proof can be extended to bigger d in a natural way.

It follows that if $\Sigma_{2,2}$-Frege enjoyed feasible interpolation, there would be an algorithm for solving mean-payoff games in polynomial time: given a game $\mathcal{G}$, run the interpolation algorithm fed with a refutation of the reflection principle formula $\mathrm{SAT} \wedge \mathrm{REF}$ and x set to the encoding of $\mathcal{C}$ from Theorem 2. Of course we let n and m be the number of variables and clauses of $\mathcal{C}$, and r and d to be the size of the Σ_2-Frege proof of $\mathcal{C}$ and 2. By Theorem 2 exactly one of $\mathrm{SAT}(\mathcal{C}, y)$ or $\mathrm{REF}(\mathcal{C}, z)$ is satisfiable, which means that the interpolation algorithm returns the other. This tells us whether $\nu_{\mathcal{G}} \geq 0$ or $\nu_{\mathcal{G}} < 0$.

Corollary 1. *There is a polynomial-time reduction from MPG to the weak automatizability of* Σ_2*-Frege, and to the interpolation problem of* $\Sigma_{2,2}$*-Frege.*

Acknowledgments. We thank Manuel Bodirsky for bringing [20] to our attention.

References

1. Alekhnovich, M., Razborov, A.A.: Resolution is not automatizable unless W[P] is tractable. In: 42nd IEEE Symp. Found. of Comp. Sci., pp. 210–219 (2001)
2. Atserias, A., Bonet, M.L.: On the automatizability of resolution and related propositional proof systems. Information and Computation 189(2), 182–201 (2004)

3. Atserias, A., Bonet, M.L., Esteban, J.L.: Lower bounds for the weak pigeonhole principle and random formulas beyond resolution. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 136–152. Springer, Heidelberg (2001)
4. Atserias, A., Maneva, E.: Mean payoff-games and the max-atom problem. Technical report (2009), http://www.lsi.upc.edu/~atserias
5. Beame, P., Pitassi, T.: Simplified and improved resolution lower bounds. In: 37th IEEE Symp. Found of Comp. Sci., pp. 274–282 (1996)
6. Ben-Sasson, E., Wigderson, A.: Short proofs are narrow–resolution made simple. J. ACM 48(2), 149–169 (2001); Prelim. version in STOC 1999
7. Bezem, M., Nieuwenhuis, R., Rodríguez-Carbonell, E.: The max-atom problem and its relevance. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 47–61. Springer, Heidelberg (2008)
8. Björklund, H., Vorobyov, S.: Combinatorial structure and randomized subexponential algorithms for infinite games. Theor. Comp. Sci. 349(3), 347–360 (2005)
9. Bonet, M., Pitassi, T., Raz, R.: Lower bounds for cutting planes proofs with small coefficients. J. Symb. Logic 62(3), 708–728 (1997); Prelim. version in STOC 1995
10. Bonet, M.L., Domingo, C., Gavaldà, R., Maciel, A., Pitassi, T.: Non-automatizability of bounded-depth Frege proofs. Computational Complexity 13, 47–68 (2004); Prelim. version in CCC 1999
11. Bonet, M.L., Pitassi, T., Raz, R.: On interpolation and automatization for Frege systems. SIAM J. Comp. 29(6), 1939–1967 (2000); Prelim. version in FOCS 1997
12. Ehrenfeucht, A., Mycielsky, J.: Positional strategies for mean payoff games. International Journal of Game Theory 8(2), 109–113 (1979)
13. Furst, M., Saxe, J., Sipser, M.: Parity, circuits and the polynomial time hierarchy. Mathematical Systems Theory 17, 13–27 (1984)
14. Jurdziński, M.: Deciding the winner in parity games is in UP ∩ co-UP. Information Processing Letters 68, 119–124 (1998)
15. Jurdziński, M., Paterson, M., Zwick, U.: A deterministic subexponential algorithm for solving parity games. SIAM J. Comp. 38(4), 1519–1532 (2008)
16. Karp, R.M.: A characterization of the minimum cycle mean in a digraph. Discrete Mathematics 23(3), 309–311 (1978)
17. Krajícek, J.: Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. J. Symb. Logic 62, 457–486 (1997)
18. Krajícek, J., Pudlák, P.: Some consequences of cryptographical conjectures for S_2^1 and EF. Information and Computation 140(1), 82–94 (1998)
19. Lovász, L., Plummer, M.D.: Matching Theory. AMS Chelsea Publ. (2009)
20. Möhring, R.H., Skutella, M., Stork, F.: Scheduling with AND/OR Precedence Constraints. SIAM J. Comp. 33(2), 393–415 (2004)
21. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading (1995)
22. Pitassi, T., Santhanam, R.: Effectively polynomial simulations. In: Proceedings of First Symposium on Innovations in Computer Science, ICS (2010)
23. Pudlák, P.: Lower bounds for resolution and cutting plane proofs and monotone computations. J. Symb. Logic 62(3), 981–998 (1997)
24. Pudlák, P.: On the complexity of the propositional calculus. In: Sets and Proofs, Invited Papers from Logic Colloq. 1997, pp. 197–218. Cambridge Univ. Press, Cambridge (1999)
25. Pudlák, P.: On reducibility and symmetry of disjoint NP-pairs. Theor. Comp. Sci. 295, 323–339 (2003)
26. Pudlák, P., Sgall, J.: Algebraic models of computation and interpolation for algebraic proof systems. In: Proof Complexity and Feasible Arithmetic. AMS (1998)
27. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theor. Comp. Sci. 158, 343–359 (1996)

Online Network Design with Outliers

Aris Anagnostopoulos[1], Fabrizio Grandoni[2], Stefano Leonardi[3], and Piotr Sankowski[4]

[1] Sapienza University of Rome, Rome, Italy
aris@dis.uniroma1.it
[2] University of Rome Tor Vergata, Rome, Italy
grandoni@disp.uniroma2.it
[3] Sapienza University of Rome, Rome, Italy
leon@dis.uniroma1.it
[4] Sapienza University of Rome, Rome, Italy
and University of Warsaw, Warsaw, Poland
sankowski@dis.uniroma1.it

Abstract. In a classical online network design problem, traffic requirements are gradually revealed to an algorithm. Each time a new request arrives, the algorithm has to satisfy it by augmenting the network under construction in a proper way (with no possibility of recovery). In this paper we study a natural generalization of the problems above, where a fraction of the requests (the *outliers*) can be disregarded. Now, each time a request arrives, the algorithm first decides whether to satisfy it or not, and only in the first case it acts accordingly; in the end at least k out of t requests must be selected. We cast three classical network design problems into this framework, the *Online Steiner Tree with Outliers*, the *Online TSP with Outliers*, and the *Online Facility Location with Outliers*.

We focus on the known distribution model, where terminals are independently sampled from a given distribution. For all the above problems, we present bicriteria online algorithms that, for any constant $\epsilon > 0$, select at least $(1-\epsilon)k$ terminals with high probability and pay in expectation $O(\log^2 n)$ times more than the expected cost of the optimal offline solution (selecting k terminals). These upper bounds are complemented by inapproximability results.

1 Introduction

In a classical *online network design* problem, traffic requirements are revealed gradually to an algorithm. Each time a new request arrives, the algorithm has to satisfy it by augmenting the network under construction in a proper way. An online algorithm is α-*competitive* (or α-*approximate*) if the ratio between the solution computed by the algorithm and the optimal (offline) solution is at most α.

For example, in the *Online Steiner Tree* problem (OST), we are given an n-node graph $G = (V, E)$, with edge weights $c : E \to \mathbb{R}^+$, and a root node r. Then t terminal nodes (where t is known to the algorithm) arrive one at a time. Each time a new terminal arrives, we need to connect it to the Steiner tree $\mathcal{S}$ under construction (initially containing the root only), by adding a proper set of edges to the tree. The goal is minimizing the final cost of the tree. The input for the *Online TSP* problem (OTSP) is the same as in OST. The difference is that here the solution is a permutation ϕ of the input

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 114–126, 2010.

terminals. (Initially, $\phi = (r)$). Each time a new terminal arrives, we can insert it into ϕ at an arbitrary point. The goal is to minimize the length of shortest cycle visiting the nodes in ϕ according to their order of appearance in ϕ. In the *Online Facility Location* problem (OFL), we are also given a set of facility nodes $\mathcal{F}$, with associated opening costs $o : V \to \mathbb{R}^+$. Now, each time a new terminal v arrives, it must be connected to some facility f_v: f_v is opened if not already the case. The goal is to minimize the facility location cost given as $\sum_{e \in F} f(e) + \sum_{v \in K} dist_G(v, e_v)$, where $F = \cup_{v \in K} f_v$ is the set of open facilities[1].

When the input sequence is chosen by an adversary, $O(\log n)$-approximation algorithms are known for the problems above, and this approximation is tight [16,26,28]. Recently, the authors of [13] studied the case where the sequence of terminals is sampled from a given distribution. For these relevant special cases, they provided online algorithms with $O(1)$ expected competitive ratio[2]. This shows a logarithmic approximability gap between worst-case and stochastic variants of online problems.

Stochastic Online Network Design with Outliers. In this paper we study a natural generalization of online network design problems, where a fraction of the requests (the *outliers*) can be disregarded. Now, each time a request arrives, the algorithm first decides whether to satisfy it or not, and only in the first case updates the network under construction accordingly. Problems with outliers have a natural motivation in the applications. For example, mobile phone companies often declare the percentage of the population which is covered by their network of antennas. In order to declare a large percentage (and attract new clients), they sometimes place antennas also in areas where costs exceed profits. However, covering everybody would be too expensive. One option is choosing some percentage of the population (say, 90%), and covering it in the cheapest possible way. This type of problems is well-studied in the offline setting, but it was never addressed before in the online case (to the best of our knowledge).

We restrict our attention to the outlier version of the three classical online network design problems mentioned before: *Online Steiner Tree with Outliers* (outOST), *Online TSP with Outliers* (outOTSP), and *Online Facility Location with Outliers* (outOFL). For each such problem, we assume that only $0 < k < t$ terminals need to be connected in the final solution.

It is easy to show that, for $k \leq t/2$, the problems above are not approximable in the adversarial model. The idea is providing k terminals with connection cost $M \gg k$. If the online algorithm selects at least an element among them, the next elements have connection cost 0. Otherwise, the next elements have connection cost M^2 and the online algorithm is forced to pay a cost of kM^2. Essentially the same example works also if we allow the online algorithm to select only $(1 - \epsilon)k \geq 1$ elements. For this reason and following [13], from now on we focus our attention on the *stochastic* setting,

[1] For a weighted graph G, $dist_G(u, v)$ denotes the distance between nodes u and v in the graph. For the sake of simplicity, we next associate an infinite opening cost to nodes which are not facilities, and let $\mathcal{F} = V$.

[2] Throughout this paper the expected competitive ratio, also called ratio of expectations (RoE), is the ratio between the expected cost of the solution computed by the online algorithm considered and the expected cost of the optimal offline solution. Sometimes in the literature the expectation of ratios EoR is considered instead (which is typically more involving).

where terminals are sampled from a given probability distribution[3]. As we will see, these stochastic online problems have strong relations with classical secretary problems.

There are two models for the stochastic setting: the *known-distribution* and the *unknown-distribution* models. In the former the algorithm knows the distribution from which terminals are sampled. In the latter the algorithm does not have any information about the distribution apart from the incoming online requests.

Our Results and Techniques. First, we give inaproximability results and lower bounds. For the known-distribution model we show that the considered problems are inapproximable if we insist on selecting exactly k elements, for $k = 1$ and for $k = t - 1$. To prove these results we need to carefully select input distributions that force the online algorithm to make mistakes: if it decides to select a terminal then with sufficiently high probability there will be cheap subsequent requests, inducing a large competitive ratio, while if it has not selected enough terminals it will be forced to select the final terminals, which with significant probability will be costly.

Furthermore, we prove an $\Omega(\log n/\log\log n)$ lower bound on the expected competitive ratio even when the online algorithm is allowed to select αk terminals only, for a constant $\alpha \in (0, 1)$. To prove it we use results from urn models.

Finally, for the unknown-distribution model we show a lower bound of $\Omega(\log n)$ for $k = \Theta(t)$ if the online algorithm is required to select $k - O(k^\alpha)$ requests for $0 \leq \alpha < 1$.

Given the inapproximability results for the case that the online algorithm has to select exactly k terminals, we study bicriteria algorithms, which select, for any given $\epsilon > 0$, at least $(1 - \epsilon)k$ terminals with high probability[4], and pay in expectation $O(\log^2 n)$ times more than the expected cost of the optimal offline solution (selecting at least k terminals).

To obtain these results, we are first able to show that very simple algorithms provide a $O(k)$ expected competitive ratio. Henceforth, the main body of the paper is focused on the case $k = \Omega(\log n)$. Our algorithms crucially exploit the probabilistic embeddings of graph metrics into tree metrics developed by Bartal et al. [4,9]. A Bartal tree of the input graph is used to partition the nodes into a collection of groups of size $\Theta(\frac{n}{t}\log n)$. Note that $\Theta(\log n)$ terminals are sampled in each group with high probability. Next, in the case of the outOST problem, we compute an anticipatory solution formed by a Steiner tree on k out of t terminals sampled beforehand from the known distribution. The anticipatory solution is deployed by the algorithm. When the actual terminals arrive, the algorithm selects all terminals that belong to a group (which we *mark*) that contains at least one terminal selected in the anticipatory solution, and connects the selected terminals to the anticipatory solution itself. Roughly speaking, there are $\Theta(k/\log n)$ marked groups and each such group collects $\Theta(\log n)$ actual terminals: altogether, the number of connected terminals is $\Theta(k)$. A careful charging argument shows that the connection cost to the anticipatory solution is in expectation $O(\log n)$ times the cost of the embedding of the anticipatory solution in the Bartal tree. In expectation, this tree embedding costs at most $O(\log n)$ times more than the anticipatory solution itself, which in turn

[3] For the sake of shortness, we will drop the term stochastic from problem names.

[4] Throughout this paper we use the term *with high probability* (abbreviated whp.) to refer to probability that approaches 1 as k, the number of selected terminals, grows. In particular, the probability of failure is polynomially small for $k = \Omega(\log n)$ in the cases considered.

costs $O(1)$ times more than the optimal solution. Altogether, this gives a $O(\log^2 n)$ competitive ratio.

The results on outOST immediately generalize to the case of outOTSP, modulo constant factors: for this reason we will describe the results for outOST only. The basic idea is to construct a Steiner tree using an online algorithm for outOST, and to duplicate its edges. This defines a multi-graph containing an Euler tour spanning $\Theta(k)$ terminals. By shortcutting the Euler tour we obtain the desired permutation ϕ of selected terminals. In each step the Euler tour can be updated preserving the the relative order of the terminals in the permutation. The cost of the optimal Steiner tree is a lower bound on the cost of the optimal TSP tour. Edge duplication introduces only a factor 2 in the approximation. Summarizing the discussion above.

Lemma 1. *Given an online α-approximation algorithm for outOST, there is an online 2α-approximation algorithm for outOTSP.*

The situation for outOFL is more involved, as in addition to the connection cost we need to take care of the facilities' cost. In this case, as well, we deploy an anticipatory solution on k out of t terminals sampled beforehand from the known distribution. In order to be able to apply some charging arguments we create a new virtual metric space, which can also capture the cost of opening the facilities: we connect every vertex of the graph to a virtual root in the tree metric with an edge of cost equal to the corresponding facility opening cost. An additional complication is to decide when to open facilities that are not opened in the anticipatory solution. We open a new facility if a selected vertex is connected to the closest facility in the anticipatory solution through a path that traverses the root in the tree embedding.

To summarize our results:

- We give inapproximability results and lower bounds for the known-distribution model.
- We give $O(\log^2 n)$ approximation algorithms for the outOST (Section 3), the outOTSP, and the outOFL (Section 4) problems for the known distribution model. In the case that $k = \Theta(t)$ we give $O(\log n \log\log n)$ approximations (details will appear in the full version of the paper).
- We extend the upper and lower bounds to the unknown-distribution model (details will appear in the full version).

The problems that we consider in this paper include as a special case *minimization* versions of the *secretary* problem. In the classical *secretary* problem a set of t elements (the *secretaries*), each one with an associated non-negative numerical value, are presented one by one to the algorithm (the *employer*). The algorithm has to decide when to stop and select the current element with the goal of maximizing the value of the selected element. A well-known extension of the problem above is the *multiple-choice secretary* problem, where the algorithm has to select $k < t$ elements of the sequence with the goal of maximizing the sum of the k selected values (or, alternatively, the ranks of the selected elements). While this problem dates back to the fifties, it has recently attracted a growing interest given its connections to selecting winners in online auctions [2,15].

In the classical secretary problem, it is easy to achieve a constant approximation to the optimal expected value; for example, waiting until seeing half the elements and

then selecting the first element that has value higher than the maximum of the first half achieves in expectation a value that is at least $1/4$ of the optimal offline value. Here we show that the minimization version is strictly harder, the reason being that a wrong choice might be very costly. The hardness arises from the fact that at least k secretaries must be hired: Intuitively, if $k - x$ secretaries have been hired after $t - x$ secretaries have been sampled, the last x secretaries must be hired irrespectively of their values. So, in Theorem 2 we show that even in the simple case that $k = 1$ the cost of the online algorithm can be exponentially larger than the optimal offline cost.

For the same reason (that a wrong choice can be very costly) the online network design problems with outliers are in general strictly harder than the versions without outliers. For example, in [13] the authors show that for the known distribution model the expected ratio of the online Steiner tree problem (without outliers, corresponding to the case that $k = t$) is constant. Instead, in Theorem 1 we show that even if we let $k = t - 1$ the approximation ratio can be arbitrarily large.

Throughout this paper we use *OPT* to denote the optimal offline solution, and *opt* to denote its expected cost. For a set of elements A and a cost function c defined on such elements, $c(A) := \sum_{a \in A} c(a)$. For a graph A, we use $c(A)$ as a shortcut for $c(E(A))$.

Related work. Competitive analysis of online algorithms has a long history (e.g., [5,10,29] and the many references therein). Steiner tree, TSP, and facility location can be approximated up to a worst-case $\Theta(\log n)$ competitive factor in the online case [16,26,28]. There have been many attempts to relax the notion of competitive analysis for classical list-update, paging and k-server problems (see [5,10,17,18,24,27,30]).

In many of the online problems studied in the literature, and in particular the versions of the online problems we study here without outliers ($k = t$), the case of known distribution was easy. As we mentioned, in this case outOST, outOTSP and outOFL reduce to the online stochastic version of Steiner tree, TSP, and facility location, for which the ratio between the the expected online cost and the expected optimal cost are constant for the known distribution model [13]. In the random permutation model, Meyerson [26] shows for facility location an algorithm with $O(1)$ ratio between the expected online cost and the expected optimal cost. In the Steiner tree problem the $\Omega(\log n)$ lower bound is still retained in the random permutation model [13].

The offline versions of the problems considered here are known as the *Steiner Tree problem with Outliers* (outST), the *TSP problem with Outliers* (outTSP) and the *Facility Location problem with Outliers* (outFL). For these problems, worst-case constant approximation algorithms are known [7,12][5]. We will exploit such (offline) approximation algorithms as part of our online algorithms.

As we mentioned, the problems that we study in this paper have strong relations with the secretary problem. Secretary problems have been studied under several models. There is a rich body of research on secretary problems and the determination of optimal stopping rules in the random permutation model since the early sixties [8,11,14,25]. In this classical model a set of t arbitrary numerical values is presented to the algorithm in random order. A strategy is known that selects the best secretary with probability

[5] The k-MST problem studied in [12] and the Steiner tree problem with outliers are equivalent approximation-wise, modulo constant factors. The same holds for TSP with outliers.

$1/e$ [25]. For the multiple-choice secretary problem it has recently proposed [21] a strategy that achieves a $1 - O(\sqrt{1/k})$ fraction of the sum of the k largest elements in the sequence, i.e., a competitive ratio [5] that approaches 1 for $k \to \infty$.

In the known-distribution model, the numerical values are identical independent samples from a known distribution (e.g., [14,20]). These problems are also known as house-selling problems (e.g., [19]), and generalizations have appeared under the name of dynamic and stochastic knapsack problems [1,22]. In this model an online algorithm that maximizes the expected revenue is obtained through dynamic programming even for the multiple-choice version of the problem [14].

Secretary problems with an underlying graph structure have been recently studied in the context of online matching problems and their generalizations [3,23]. One can define a minimization version of all the problems above, as we do here. Minimization secretary problems are much less studied in the literature, and most studies cover some basic cases. In particular, researchers have studied the problems where the goal is to minimize the expected rank of the selected secretary (as opposed to the actual expected cost) or to minimize the expected cost if the input distribution is uniform in $[0, 1]$ (look, for example, the work of Bruce and Ferguson [6] and the references therein). However, to our knowledge, there has not been any comparison of the online and offline solutions for arbitrary input distributions. In particular, nothing to our knowledge was known on the gap between their costs.

2 Lower Bounds

Let us start by proving inapproximability results for outOST in the known distribution model, when we insist on selecting *exactly* k terminals. Similar lower bounds hold for outOTSP and outOFL. The proof of the following theorem will appear in the full version of the paper.

Theorem 1. *In the known distribution model, the expected competitive ratio for outOST can be arbitrarily large for $k = t - 1$.*

The next theorem considers the somehow opposite case that k is very small (proof omitted). Note that the construction in the proof shows that the minimization version of the secretary problem has an exponential competitive ratio.

Theorem 2. *In the known distribution model, the expected competitive ratio for outOST can be exponentially large in the number n of nodes for $t = 3n/4$ and $k = 1$.*

Next we present an $O(\frac{\log n}{\log \log n})$ lower bound for outOST, outOTSP and outOFL, which applies also to the case that the online algorithm is allowed to connect only $\alpha\, k$ terminals, for a sufficiently large constant $\alpha \in (0, 1)$.

Theorem 3. *Assume that an online algorithm for outOST (resp., outOTSP or outOFL) is allowed to connect αk terminals, for a sufficiently large constant $\alpha \in (0, 1)$. Then the expected competitive ratio is $\Omega\left(\frac{\log n}{\log \log n}\right)$.*

Proof. We give the proof for outOST. The proof for the other two problems is analogous. Consider the star graph with the root r as center, and uniform edge weights 1.

(Preprocessing Phase)

Step 1. Compute a Bartal tree $\mathcal{B}$ for the input graph. Partition the leaves of $\mathcal{B}$ from left to right in groups $V_1, \ldots, V_{n/\sigma}$ of size σ.

Step 2. Sample t nodes $\tilde{T}$ from the input probability distribution. Compute a ρ_{outST}-approximate solution $\tilde{\mathcal{S}}$ to the (offline) outST problem induced by $\tilde{T}$. Let $\tilde{K}$ be the resulting set of k terminals, and $\mathcal{K}$ be the nodes of groups with at least one node in $\tilde{K}$, excluding the leftmost and rightmost such groups. Set $\mathcal{S} = \tilde{\mathcal{S}}$.

(Online Phase)

Step 3. For each input node $v \in T$, if $v \in \mathcal{K}$, add v to K and augment $\mathcal{S}$ with a shortest path to v.

Fig. 1. Algorithm `outost-large` for outOST

Suppose that each leaf is sampled with uniform probability $1/(n-1)$. Let $t = n-1$ and $k = \frac{\ln n}{c \ln \ln n}$, for a sufficiently large constant c. When a leaf is sampled at least k times, the optimum solution cost is 1, and in any case it is not larger than $n-1$. By standard balls-and-bins results, the probability that no leaf is sampled at least k times is polynomially small in n. Hence $opt = O(1)$.

Take now any online algorithm. Suppose that at some point this algorithm connects a terminal v for the first time. After this choice, the same terminal v will be sampled $O(1)$ times in expectation. Hence, the expected total number of connected terminals is proportional to the number of distinct leaves which are connected. This implies that the online algorithm is forced to connect $\Omega(k)$ distinct nodes in expectation, with a cost of $\Omega(k)$. Therefore, the competitive ratio is $\Omega(k) = \Omega\left(\frac{\log n}{\log \log n}\right)$. □

We observe that the proof above applies to the case of small values of k. Extending the proof to large values of k (or finding a better algorithm in that case) is an interesting open problem. The next theorem (proof omitted) moves along these lines.

Theorem 4. *In the unknown distribution model, the expected competitive ratio for outOST is $\Omega(\log n)$ if the online algorithm is required to connect $(1-\epsilon)k$ terminals for $\epsilon < \frac{\log n}{\sqrt{n}}$, $t = n$ and $k = \frac{t}{2}$.*

3 Online Steiner Tree with Outliers

In this section we consider the Online Steiner Tree problem with Outliers (outOST). We consider the case that the input distribution is the uniform distribution, while the generalization for any distribution will appear in the full version of the paper.

First, let us assume $k \geq c \log n$ for a large enough constant $c > 0$. We next describe an algorithm `outost-large` with $O(\log^2 n)$ competitive ratio, which connects at least $(1-\epsilon)k$ terminals with high probability, for any given constant parameter $\epsilon > 0$.

A crucial step is constructing a Bartal tree $\mathcal{B}$ over the input graph G using the algorithm in [9]. We recall that $\mathcal{B} = (W, F)$ is a rooted tree, with edge costs $c_{\mathcal{B}} : F \to \mathbb{R}^+$, whose leaves are the nodes V, and such that the following two properties hold:

1. Edges at the same level in the tree have the same cost and given edges e and f at level i and $i+1$, respectively (the root is at level zero), $c_{\mathcal{B}}(e) = 2c_{\mathcal{B}}(f)$.
2. For any two leaves $u, v \in \mathcal{B}$, $\frac{1}{O(\log n)} E[dist_{\mathcal{B}}(u,v)] \leq dist_G(u,v) \leq dist_{\mathcal{B}}(u,v)$.

Algorithm `outost-large` is described in Figure 1. The algorithm starts with two preprocessing steps. Initially it computes a Bartal tree $\mathcal{B}$ for G, and partitions its leaves from left to right into groups $V_1, V_2, \ldots, V_{n/\sigma}$ of size $\sigma = \alpha \frac{n}{t} \log n$ each, for a constant α to be fixed later[6]. Then the algorithm samples t nodes $\tilde{T}$, and constructs a Steiner tree $\tilde{\mathcal{S}}$ (*anticipatory solution*) on k such nodes $\tilde{K}$, using a $\rho_{outST} = O(1)$ approximation algorithm for (offline) outST [12][7]. We call *azure* and *blue* the nodes in $\tilde{T}$ and $\tilde{K}$, respectively. We also call *blue* the groups containing at least one blue node, and *boundary* the leftmost and rightmost blue groups. The Steiner tree $\mathcal{S}$ under construction is initially set to $\tilde{\mathcal{S}}$.

In the online part of the algorithm, each time a new terminal $v \in T$ arrives, v is added to the set K of selected terminals if and only if v belongs to a non-boundary blue group. In that case, the algorithm also adds to $\mathcal{S}$ a shortest path from v to $\mathcal{S}$. We call *orange* and *red* the nodes in T and K, respectively. It turns out that the connection of orange nodes in blue groups can be conveniently charged to the cost of the anticipatory solution (boundary blue groups are excluded for technical reasons).

Let us initially bound the number of red nodes, that is, the number of terminals connected by the algorithm.

Lemma 2. *For any $\epsilon > 0$ and $\sigma = \alpha \frac{n}{t} \log n$, there is a choice of $\alpha > 0$ such that the number of red nodes is at least $(1-\epsilon)k$ with high probability.*

Proof. The number N_i of azure (resp., orange) nodes in a given group V_i, counting repetitions, satisfies $\mathrm{E}[N_i] = \frac{t}{n}\frac{n}{t}\alpha \log n = \alpha \log n$. Let $\delta \in (0,1)$ be a sufficiently small constant. By Chernoff's bounds, we know that there is a value of $\alpha > 0$ such that the probability of the event $\{N_i \notin [(1-\delta)\alpha \log n, (1+\delta)\alpha \log n]\}$ is smaller than any given inverse polynomial in n. Hence, from the union bound, with high probability all the groups contain between $(1-\delta)\alpha \log n$ and $(1+\delta)\alpha \log n$ azure (resp., orange) nodes. Let us assume from now on that this event happens. Recall that by assumption $k \geq c \log n$ for a sufficiently large constant $c > 0$.

Each blue group contains at most $(1+\delta)\alpha \log n$ azure (and hence blue) nodes. Therefore, there are at least $\frac{k}{(1+\delta)\alpha \log n}$ blue groups, and so the number of orange nodes in non-boundary blue groups (i.e. the number of red nodes) is at least

$$(1-\delta)\alpha \log n \left(\frac{k}{(1+\delta)\alpha \log n} - 2 \right) \geq \frac{1-\delta}{1+\delta} k - 2\frac{(1-\delta)\alpha}{c} k.$$

The latter quantity is at least $(1-\epsilon)k$ for proper constants c and δ. □

We continue by proving the following basic tool lemma that will be reused for outOFL later on. Refer to Figure 2. Let r_v (resp., ℓ_v) be the first blue node to the right (resp.,

[6] To avoid inessential technicalities, we will always assume that n is a multiple of σ.

[7] Since the cost of an MST spanning a set of vertices W is at most twice the corresponding cost of the best Steiner tree connecting those vertices, we can obtain a constant approximation for the outST problem if we have a constant approximation for the k-MST problem.

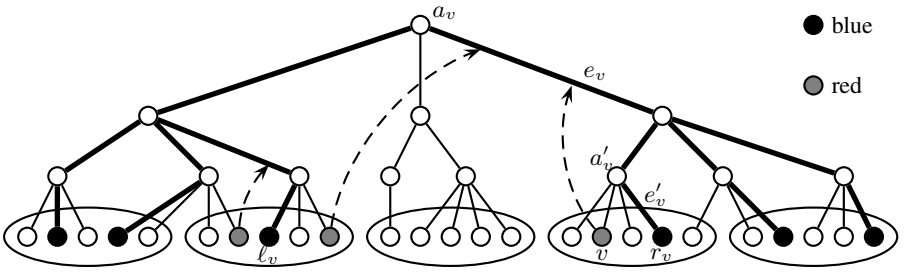


Fig. 2. Charging scheme in the analysis of `outost-large`. Bold edges indicate the subtree $\tilde{\mathcal{B}}$. Groups are enclosed into ellipses. Dashed arcs reflect the charging of red nodes connections to the edges of $\tilde{\mathcal{B}}$.

left) of node $v \in K$ (with respect to the given ordering of leaves from left to right). Note that r_v and ℓ_v are well defined, since the boundary blue groups are not used to define K.

Lemma 3. *Let $\tilde{\mathcal{B}}$ be any subtree in $\mathcal{B}$ spanning nodes in $\tilde{K}$. Then*

$$\mathrm{E}\left[\sum_{v\in K} dist_{\mathcal{B}}(v, r_v)\right] \leq 8\sigma\tfrac{t}{n}\,\mathrm{E}[c_{\mathcal{B}}(\tilde{\mathcal{B}})].$$

Proof. The idea of the proof it to charge the distances $dist_{\mathcal{B}}(v, r_v)$ to a proper subset of edges $\tilde{E} \subseteq E(\tilde{B})$, so that each such edge is charged $O(\sigma\frac{t}{n})$ times in expectation. Let a_v (resp., a'_v) be the lowest common ancestor of ℓ_v (resp., v) and r_v. Let moreover e_v (resp., e'_v) be the first edge along the path from a_v (resp., a'_v) to r_v. (See also Figure 2). Since v lies between ℓ_v and r_v, the level of a'_v is not higher than the level of a_v. We can conclude by Property 1 of Bartal trees that $c_{\mathcal{B}}(e'_v) \leq c_{\mathcal{B}}(e_v)$. Property 1 also implies that $dist_{\mathcal{B}}(v, r_v) = dist_{\mathcal{B}}(v, a'_v) + dist_{\mathcal{B}}(a'_v, r_v) \leq 4c_{\mathcal{B}}(e'_v)$. Altogether, we obtain

$$\mathrm{dist}_{\mathcal{B}}(v, r_v) \leq 4c_{\mathcal{B}}(e_v). \tag{1}$$

Let $\tilde{E} := \cup_{v\in K} e_v \subseteq E(\tilde{\mathcal{B}})$. Consider any edge $e = e_w \in \tilde{E}$. Any red node u to the left of ℓ_w or to the right of r_w satisfies $e_u \neq e_w$. We conclude that the set $\tilde{V}_e := \{v \in K : e_v = e\}$ is a subset of the red nodes contained in the groups of r_w and ℓ_w. Then

$$\mathrm{E}\left[\sum_{v\in K} c_{\mathcal{B}}(e_v)\right] = \mathrm{E}\left[\sum_{e\in\tilde{E}} |\tilde{V}_e| \cdot c_{\mathcal{B}}(e)\right] \leq 2\sigma\frac{t}{n}\,\mathrm{E}\left[\sum_{e\in\tilde{E}} c_{\mathcal{B}}(e)\right] \leq 2\sigma\frac{t}{n}\mathrm{E}\left[c_{\mathcal{B}}(\tilde{\mathcal{B}})\right]. \tag{2}$$

The lemma follows by summing up over v the expectation of (1) and combining it with (2). □

We are now ready to bound the competitive ratio of the algorithm.

Lemma 4. *The expected cost of the solution computed by algorithm* `outost-large` *is* $O(\sigma \frac{t}{n} \log n)$ *times the expected cost of the optimum offline solution.*

Proof. The anticipatory problem instance is sampled from the same distribution as the real problem instance, so $\mathrm{E}[c(\tilde{\mathcal{S}})] \leq \rho_{outST} \cdot opt = O(opt)$.

Let us bound the cost C_{on} paid by the algorithm during the online phase. Consider the minimal subtree $\tilde{\mathcal{B}}$ of $\mathcal{B}$ spanning $\tilde{K} \cup \{r\}$. Of course, $\tilde{\mathcal{B}}$ is an optimal Steiner tree over $\tilde{K} \cup \{r\}$ with respect to graph $\mathcal{B}$. It follows from Property 2 and the fact that the cost of a minimum spanning tree is twice the cost of a Steiner tree that connects the same vertices that

$$\mathrm{E}[c_{\mathcal{B}}(\tilde{\mathcal{B}})] \leq \mathrm{E}[2\, O(\log n) c(\tilde{\mathcal{S}})] = O(\log n) \cdot opt. \tag{3}$$

We have

$$C_{on} \leq \sum_{v \in K} dist_G(v, \tilde{K}) \overset{\text{Prop. 2}}{\leq} \sum_{v \in K} dist_{\mathcal{B}}(v, \tilde{K}) \leq \sum_{v \in K} dist_{\mathcal{B}}(v, r_v). \tag{4}$$

$\tilde{\mathcal{B}}$ satisfies the conditions of Lemma 3, hence by putting everything together we obtain

$$\mathrm{E}[C_{on}] \overset{(4)}{\leq} \mathrm{E}\left[\sum_{v \in K} dist_{\mathcal{B}}(v, r_v)\right] \overset{\text{Lem. 3}}{\leq} 8\sigma \frac{t}{n} \mathrm{E}[c_{\mathcal{B}}(\tilde{\mathcal{B}})] \overset{(3)}{=} O\left(\sigma \frac{t}{n} \log n\right) \cdot opt. \qquad \square$$

Note that up to now we have assumed that $k = \Omega(\log n)$. The following simple algorithm, `outost-small`, has competitive ratio $O(k)$ (proof omitted), so it can be applied in the case that $k = O(\log n)$.

Let W be the set of the $(1-\delta)\frac{n}{t}k$ nodes which are closest to the root (breaking ties arbitrarily). Here $\delta \in (0,1)$ is a proper constant. Whenever a new node $v \in T$ arrives, `outost-small` adds it to the set K of selected nodes iff $v \in W$. In that case, the algorithm connects v to the current tree $\mathcal{S}$ via a shortest path.

Let `outost` be the (polynomial-time) algorithm for outOST which either runs `outost-small` for $k < c \log n$, or `outost-large` with $\sigma = \alpha \frac{n}{t} \log n$ otherwise. T he following theorem easily follows from Lemmas 2 and 4.

Theorem 5. *For any given* $\epsilon > 0$ *and for* $\sigma = \alpha \frac{n}{t} \log n$*, Algorithm* `outost` *connects at least* $(1-\epsilon)k$ *terminals with high probability. The expected cost of the solution is* $O\left(\sigma \frac{t}{n} \log n\right) = O(\log^2 n)$ *times the expected cost of the optimum offline solution.*

4 Online Facility Location with Outliers

In this section we consider the Online Facility Location problem with Outliers (outOFL). Like in the case of outOST, let us assume that $k \geq c \log n$ for a sufficiently large constant $c > 0$, while again a simple algorithm can handle the case that $k \leq c \log n$.

Our algorithm `outofl-large` is described in Figure 3. Let $G_r = (V \cup r, E')$ be a graph obtained from G by adding a new vertex r and connecting it to all other vertices v with edges of cost $\mathrm{o}(v)$. We denote by c_{G_r} the edge weights of G_r. Note that every facility location solution $\mathcal{F} = (F, K)$ in G can be mapped to a Steiner tree $T_{\mathcal{F}}$ in G_r spanning $K \cup \{r\}$ with the same cost: it is sufficient to augment the connection paths

(Preprocessing Phase)

Step 1. Construct the graph G_r and compute a Bartal tree $\mathcal{B}$ for G_r. Partition the leaves of $\mathcal{B}$ from left to right in groups $V_1, \ldots, V_{n/\sigma}$ of size σ.

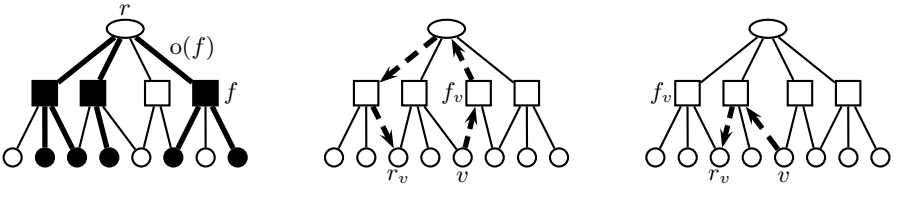
Step 2. Sample t nodes $\tilde{T}$ from the input probability distribution. Compute a ρ_{outFL}-approximate solution $\tilde{\mathcal{F}} = (\tilde{F}, \tilde{K})$ to the (offline) facility location problem with outliers induced by $\tilde{T}$, where $\tilde{F}$ and $\tilde{K}$ are the open facilities and the selected set of k terminals, respectively. Let $\mathcal{K}$ be the nodes of groups with at least one node in $\tilde{K}$, excluding the leftmost and rightmost such groups. Open the facilities in $\tilde{F}$.

(Online Phase)

Step 3. For each input node $v \in T$, if $v \in \mathcal{K}$, add v to K. Let r_v be the first node from $\tilde{K}$ to the right of v. Consider the shortest path π from v to r_v in G_r:

- **Step 3.1.** If π goes through r, then let (f_v, u) be the first edge on π such that $u = r$. Open facility f_v, if not already open, and connect v to f_v.
- **Step 3.2.** Otherwise connect v to the facility f_v to which node r_v is connected in $\tilde{\mathcal{F}}$.

Fig. 3. Algorithm `outofl-large` for outOFL



(a) (b) (c)

Fig. 4. An example of graph G_r is given in (a). For clarity of illustration, we distinguished between terminals (circles) and facilities (squares). The oval node is the root. Terminals $\tilde{K}$ and the open facilities in the corresponding anticipatory solution are drawn in bold, as well as the associated Steiner tree $T_{\mathcal{F}}$. Examples of Steps 3.1 and 3.2 are given in (b) and (c), respectively (bold edges indicate one possible shortest path from v to r_v).

in $\mathcal{F}$ with the edges between open facilities and r. Unfortunately, solving a outOST problem on G_r is not sufficient to solve the original outOFL problem. This is because not every tree in G_r corresponds to a valid facility location solution. Nevertheless, the graph G_r is very useful in our case as it allows to introduce a convenient metric into the facility location problem. (See Figure 4 for an example of graph G_r, and a corresponding implementation of Steps 3.1 and 3.2).

First of all note that the set of nodes that are selected by the algorithm are defined in the same way as in Algorithm `outost-large`, that is, by a constant approximation to the (offline) outFL problem on a set of sampled terminals $\tilde{T}$, using, for example, the algorithm of Charikar et al. [7]. Hence, Lemma 2 holds here as well. Therefore, we only need to show that the cost of the online solution is small. The proof of the following theorem will appear in the full version of the paper.

Theorem 6. *For any given $\epsilon > 0$, Algorithm* `outofl` *connects at least $(1-\epsilon)k$ terminals with high probability. The expected cost of the solution is $O(\sigma \frac{t}{n} \log n) = O(\log^2 n)$ times the expected cost of the optimum offline solution.*

Acknowledgements. We would like to thank Anupam Gupta for fruitful discussions. This work was partially supported by the Polish Ministry of Science grant N206 355636.

References

1. Babaioff, M., Immorlica, N., Kempe, D., Kleinberg, R.: A knapsack secretary problem with applications. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) RANDOM 2007 and APPROX 2007. LNCS, vol. 4627, pp. 16–28. Springer, Heidelberg (2007)
2. Babaioff, M., Immorlica, N., Kempe, D., Kleinberg, R.: Online auctions and generalized secretary problems. SIGecom Exch. 7(2), 1–11 (2008)
3. Babaioff, M., Immorlica, N., Kleinberg, R.: Matroids, secretary problems, and online mechanisms. In: SODA 2007, Philadelphia, PA, USA. pp. 434–443, Society for Industrial and Applied Mathematics (2007)
4. Bartal, Y.: On approximating arbitrary metrics by tree metrics. In: STOC 1998: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, pp. 161–168 (1998)
5. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press, New York (1998)
6. Bruce, F.T., Ferguson, T.S.: Minimizing the expected rank with full information. Journal of Applied Probability 30(3), 616–626 (1993)
7. Charikar, M., Khuller, S., Mount, D.M., Narasimhan, G.: Algorithms for facility location problems with outliers. In: SODA 2001: Proceedings of the 12th annual ACM-SIAM symposium on Discrete algorithms, pp. 642–651 (2001)
8. Dynkin, E.B.: The optimum choice of the instant for stopping a markov process. Sov. Math. Dokl. 4 (1963)
9. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. Journal of Computer and System Sciences 69(4), 485–497 (2004)
10. Fiat, A. (ed.): Dagstuhl Seminar 1996. LNCS, vol. 1442. Springer, Berlin (1998)
11. Freeman, P.: The secretary problem and its extensions: a review. Internat. Statist. Rev. 51(2), 189–206 (1983)
12. Garg, N.: Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In: STOC 2005: Proceedings of the 37th annual ACM symposium on Theory of computing, pp. 396–402 (2005)
13. Garg, N., Gupta, A., Leonardi, S., Sankowski, P.: Stochastic analyses for online combinatorial optimization problems. In: SODA 2008, pp. 942–951 (2008)
14. Gilbert, J.P., Mosteller, F.: Recognizing the maximum of a sequence. Journal of the American Statistical Association 61(313), 35–73 (1966)
15. Hajiaghayi, M.T., Kleinberg, R., Parkes, D.C.: Adaptive limited-supply online auctions. In: EC 2004: Proceedings of the 5th ACM conference on Electronic commerce, pp. 71–80 (2004)
16. Imase, M., Waxman, B.M.: Dynamic Steiner tree problem. SIAM J. Discrete Math. 4(3), 369–384 (1991)
17. Irani, S., Karlin, A.R.: On online computation. In: Hochbaum, D. (ed.) Approximation Algorithms for NP Hard Problems. PWS publishing Co. (1996)
18. Karlin, A.R., Phillips, S.J., Raghavan, P.: Markov paging. SIAM J. Comput. 30(3), 906–922 (2000)

19. Karlin, S.: Stochastic models and optimal policy for selling an asset. In: Studies in applied probability and management science, pp. 148–158 (1962)
20. Kennedy, D.: Prophet-type inequalities for multichoice optimal stopping. Stoch. Proc. Appl. 24(1), 77–88 (1987)
21. Kleinberg, R.: A multiple-choice secretary algorithm with applications to online auctions. In: SODA 2005, pp. 630–631 (2005)
22. Kleywegt, A.J., Papastavrou, J.D.: The dynamic and stochastic knapsack problem. Oper. Res. 46(1), 17–35 (1998)
23. Korula, N., Pal, M.: Algorithms for secretary problems on graphs and hypergraphs. CoRR, abs/0807.1139 (2008)
24. Koutsoupias, E., Papadimitriou, C.H.: Beyond competitive analysis. SIAM J. Comput. 30(1), 300–317 (2000)
25. Lindley, D.V.: Dynamic programming and decision theory. Applied Statistics 10, 39–51 (1961)
26. Meyerson, A.: Online facility location. In: FOCS 2001, pp. 426–431 (2001)
27. Raghavan, P.: A statistical adversary for on-line algorithms. In: Online Algorithms. DIMACS Ser. Discrete Math. Theoret. Comput. Sci, vol. 53, pp. 79–83 (1991)
28. Rosenkrantz, D.J., Stearns, R.E., Lewis II, P.M.: An analysis of several heuristics for the traveling salesman problem. SIAM J. Comput. 6(3), 563–581 (1977)
29. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Comm. ACM 28(2), 202–208 (1985)
30. Young, N.E.: On-line paging against adversarially biased random inputs. J. Algorithms 37(1), 218–235 (2000)

Efficient Completely Non-malleable Public Key Encryption

Benoît Libert[1,⋆] and Moti Yung[2]

[1] Université catholique de Louvain, Belgium
[2] Google Inc. and Columbia University, USA

Abstract. Non-malleable encryption schemes make it infeasible for adversaries provided with an encryption of some plaintext m to compute another ciphertext encrypting a plaintext m' that is related to m. At ICALP'05, Fischlin suggested a stronger notion, called *complete non-malleability*, where non-malleability should be preserved against adversaries attempting to compute encryptions of related plaintexts under newly generated public keys. This new notion applies to systems where on-line certificate authorities are available and users can issue keys on-the-fly. It was originally motivated by the design of non-malleable commitments from public key encryption (i.e., extractable commitments), for which the usual flavor of non-malleability does not suffice. Completely non-malleable encryption schemes are known not to exist w.r.t. black-box simulation in the standard model (although constructions are possible in the random oracle model). One of the original motivations of Fischlin's work was to have non-malleable commitments without preconditions.

At PKC'08, Ventre and Visconti investigated complete non malleability as a general notion suitable for protocol design, and departed from only considering it as a tool for commitment schemes without preconditions. Indeed, if one allows members of a community to generate public keys "on the fly", then considering the notion is justified: For example, if a bidder in an auction scheme can, in the middle of the auction process, register a public key which is malleable with respect to a scheme used in an already submitted bid, he may produce a slightly higher bid without even knowing the already submitted bid. Only when the latter is opened he may be able to open its bid. In this more general context, Ventre and Visconti showed that completely non malleable schemes do exist in the standard model; in fact in the shared random string model as well as in the interactive setting. Their non-interactive scheme is, however, inefficient as it relies on the generic NIZK approach. They left the existence of efficient schemes in the common reference string model open. In this work we describe the first *efficient* constructions that are completely non-malleable in this standard model.

Keywords: Public key encryption, complete non-malleability, efficiency.

⋆ This author acknowledges the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.) for their support and the BCRYPT Interuniversity Attraction Pole.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 127–139, 2010.

1 Introduction

Introduced by Dolev, Dwork and Naor [13], the notion of non-malleable public key encryption captures the intuition that, given a public key pk and an encryption c of some message m sampled from a distribution M of her choice, a man-in-the-middle adversary $\mathcal{A}$ is not able to come up with a relation $\mathtt{R}$ and a ciphertext c' for which the underlying plaintext m' is related to m via the relation $\mathtt{R}$. Further, this task should be beyond the reach of any polynomial-time adversary having access to a decryption oracle even after having seen the challenge ciphertext. This strongest notion is usually called adaptive chosen-ciphertext non-malleability (NM-CCA2) as opposed to the non-adaptive one (NM-CCA1), where decryption queries are only permitted before the challenge phase.

Designing non-malleable schemes in the NM-CCA2 sense has proved to be hard. The initial construction of Dolev, Dwork and Naor [13] was mostly feasibility proof of concept. It was only in 1998 that a practical scheme was put forth by Cramer and Shoup [11] whose construction was later shown to fit within a general framework [12]. At the same time, the strongest notion of non-malleability (NM-CCA2) was proved equivalent [1] to that of indistinguishability against chosen-ciphertext attacks (IND-CCA2) [23]. The latter result was established using a different definition of non-malleability (sometimes called "comparison-based" definition as opposed to the "simulation-based" definition used in [13]) but the two definitions were proved equivalent [4]. Subsequently, Sahai [25] showed how to turn any semantically secure [17] encryption scheme into a NM-CCA2 (and thus IND-CCA2) one using the Naor-Yung paradigm [24] and non-malleable non-interactive zero-knowledge (NIZK) techniques. Later on, Canetti, Halevi and Katz [9] highlighted a new paradigm to derive NM-CCA2 schemes from weakly secure identity-based encryption [27,7].

More recently, Pass, shelat and Vaikuntanathan [21] put forth a new definition of non-malleability that avoids concerns arising with earlier ones when the adversary outputs invalid ciphertexts. In the setting where decryption queries are not permitted (usually referred to as the NM-CPA scenario), they also showed how to construct a NM-CPA encryption scheme from any semantically secure one without making further assumptions (in these regards, Sahai's construction additionally required the existence of tradpoor permutations because of its use of NIZK proofs). Yet, the construction of [21] is non-black-box in that it uses designated verifier NIZK proofs which in turn depend on the circuit implementing the underlying semantically secure cryptosystem. Finally, Choi *et al.* [10] showed how to bypass the use of non-black-box techniques in the construction of NM-CPA secure systems from semantically secure ones.

Complete Non-Malleability. In 2005, Fischlin defined [15] a stronger flavor of non-malleability termed *complete non-malleability*. This notion strengthens earlier definitions by additionally allowing the adversary to choose a new public key (possibly as a function of the original one and without necessarily knowing the matching private key). Her goal is now to produce an encryption (under the new public key) of a plaintext that is related to the original one according to a more general relation also taking public keys into account.

The motivation behind this enhanced notion lies in that asymmetric encryption schemes are frequently used as building blocks within higher level protocols where previous forms of non-malleability may not suffice (e.g., protocols that allow users to issue asymmetric keys on-the-fly). Also, complete non-malleability was argued in [15] to be necessary if one desires to build non-malleable commitments on top of public key encryption schemes.

Completely non-malleable public key cryptosystems unfortunately turn out to be particularly hard to construct. It was shown in [15] that, although NM-CCA2 in the usual sense, the Cramer-Shoup encryption scheme [11] (along with several other well-known ones, even under the random oracle idealization, like RSA-OAEP [3]) is not completely non-malleable even when the adversary is disallowed to make decryption queries. Moreover, Fischlin's work also ruled out the existence of provably completely non-malleable non-interactive encryption schemes w.r.t. simulation-based black-box security in the standard model. On the other hand, such efficient constructions were described [15] in the random oracle idealization [2]: for instance, RSA-OAEP can be made completely non-malleable by simply including the public key in the inputs to the random oracles. The latter result can actually be interpreted as yet another separation (on quite a natural primitive) between the random oracle methodology and the "real world" (the first such example being [8]). Fischlin's schemes can be used to implement non-malleable commitment schemes without preprocessing or a common random string, but they suffer from being limited to the idealized setting.

Recently, Ventre and Visconti [26] revisited the problem of designing and further understanding completely non-malleable encryption schemes as general tools in protocols where public keys are generated during the protocol. They first gave a game-based definition and showed how it implies Fischlin's simulation-based one in the chosen-plaintext setting (denoted NM-CPA* to distinguish it from the usual notion dubbed NM-CPA). A similar implication was also established in the CCA1 and CCA2 scenarios for a wide class of relations. While also impossible to meet in the standard model for black-box adversaries, the game-based definition turns out to be somehow more convenient to use. Assuming that a shared random string (*i.e.*, a set of parameters that remains out of the adversary's control) is available to all parties, [26] first showed how non-malleable NIZK techniques allow constructing completely non-malleable CCA2 schemes (or NM-CCA2* for short) in the standard model from any semantically secure one. This was not in contradiction with Fischlin's impossibility result that holds in the plain model (without a reference string). As another workaround to the impossibility result of [15], Ventre and Visconti also used non-black box techniques to design an interactive completely non-malleable cryptosystem.

Although Fischlin's complete non-malleability notion was initially motivated by the design of non-malleable commitments in the plain model (*i.e.*, without setup assumptions), we believe this notion to remain of interest even in the trusted parameters setting. It indeed guarantees a strong form of plaintext extractability under adversarially-generated keys, which can come handy in scenarios – such as the computational model of Dolev and Yao [14] – where new

public keys can be dynamically chosen by adversaries, or when adversaries that at some point of time try to have a correlated key with a key generated by a honest party. For example, in [18], Herzog *et al.* showed how a certain flavor of plaintext-awareness [3] (the definition of which requires senders to go through a public key registration step) allows enforcing the conditions of the Dolev-Yao model, which places restrictions on what messages an adversary is able to form. In the same spirit, complete non-malleability could serve as a first step toward a notion related to plaintext extractability under dynamically generated keys, which could provide a way to enforce Dolev-Yao-style conditions assuming certain attack scenarios and using a common reference string instead of requiring ciphertext senders to hold and register a public key.

Our Contribution. Ventre and Visconti left open the problem of *efficiently* implementing general purpose non-interactive NM-CCA2* schemes in the standard model using a reference string. Due to the use of NIZK techniques, their first construction is more of a feasibility result than a practical solution.

Using their game-based definition, we tackle their challenge and describe the first efficient systems in the common reference string model. The first one uses the Canetti-Halevi-Katz [9] paradigm and the bilinear Diffie-Hellman assumption. The second one builds on the lossy trapdoor function primitive coined by Peikert and Waters [22]. The first proposal allows very compact ciphertexts. The second scheme yields longer ciphertexts but gives evidence that, using a common reference string, complete non-malleability is efficiently achievable even under elementary number-theoretic assumptions like Decision Diffie-Hellman, the Composite Residuosity assumption or, alternatively, using worst-case lattice assumptions. The schemes are obtained via simple yet, we believe, insightful modifications of existing NM-CCA2 schemes. Simplicity in our setting is a blessing since it first ensures efficiency (which was Ventre and Visconti's challenge) and it also helps in clearly revealing the basic ingredients needed for NM scheme to become completely-NM when transposed in the trusted common string setting.

2 Background and Definitions

Throughout the paper, when S is a set, $x \xleftarrow{\$} S$ denotes the action of choosing x uniformly at random in S. By $a \in \mathsf{poly}(\lambda)$, we mean that a is a polynomial in λ while $b \in \mathsf{negl}(\lambda)$ says that b is a negligible function of λ (*i.e.*, a function that decreases faster than the inverse of any $a \in \mathsf{poly}(\lambda)$). When A is a possibly probabilistic algorithm, $A(x) \Rightarrow b$ denotes the event that A outputs the value b when fed with the input x. For a value v, $|v|$ is the bitlength of v. The symbol $\oplus$ finally stands for the bitwise exclusive OR of two equal-length strings.

2.1 Game-Based Complete Non-Malleability

A public key cryptosystem is a triple $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ where $\mathcal{G}$ is a probabilistic algorithm outputting a key pair (sk, pk) on input of a security parameter $\lambda \in \mathbb{N}$, $\mathcal{E}$ is a probabilistic algorithm that computes a ciphertext $c = \mathcal{E}_{pk}(m, r)$ from a

plaintext m and a random string r while $\mathcal{D}$ is a deterministic algorithm such that $m = \mathcal{D}_{sk}(\mathcal{E}_{pk}(m, r))$ for any (m, r) and any pair (sk, pk) produced by $\mathcal{G}$. In the common reference string model, these algorithms additionally use a set of public parameters produced by an algorithm $\mathcal{CRS}$-$\mathcal{G}$ run by a trusted party.

The definitions of complete non-malleability given in [15,26] consider *complete relations*. A complete relation $\mathtt{R}$ is an efficient algorithm that takes as input a public key pk, a plaintext m, another public key pk^*, a vector of ciphertexts $\mathbf{c}^*$ (that are encrypted under pk^*) and the vector $\mathbf{m}^*$ of underlying plaintexts. The output of $\mathtt{R}$ is a boolean value 0 or 1. In the common reference string model, the complete relation takes the reference string as additional input.

In [15], Fischlin gave a simulation-based definition that extends the original definition of non-malleability [13]. Ventre and Visconti introduced the following game-based definition which is inspired by the comparison-based one [1].

Definition 1 ([26]). *Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a public key encryption scheme and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ denote an adversary. For* atk $\in \{\text{cpa}, \text{cca1}, \text{cca2}\}$ *and $\lambda \in \mathbb{N}$, let*

$$\mathbf{Adv}^{\text{nm-atk}^*}_{\mathcal{A}}(\lambda) = \left| \Pr[\mathbf{Expt}^{\text{nm-atk}^*\text{-}0}_{\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{Expt}^{\text{nm-atk}^*\text{-}1}_{\mathcal{A}}(\lambda) \Rightarrow 1] \right|$$

where the experiments $\mathbf{Expt}^{\text{nm-atk}^*\text{-}0}_{\mathcal{A}}(\lambda)$ and $\mathbf{Expt}^{\text{nm-atk}^*\text{-}1}_{\mathcal{A}}(\lambda)$ are defined hereafter. In these notations,

$\mathbf{Expt}^{\text{nm-atk}^*\text{-}0}_{\mathcal{A}}(\lambda)$	$\mathbf{Expt}^{\text{nm-atk}^*\text{-}1}_{\mathcal{A}}(\lambda)$
$(pk, sk) \leftarrow \mathcal{G}(\lambda)$	$(pk, sk) \leftarrow \mathcal{G}(\lambda)$
$(M, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$	$(M, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$
$x \xleftarrow{\$} M$	$x, \tilde{x} \xleftarrow{\$} M$
$c = \mathcal{E}_{pk}(x, r)$ where $r \xleftarrow{\$} \{0,1\}^{\mathtt{poly}(\lambda)}$	$c = \mathcal{E}_{pk}(x, r)$ where $r \xleftarrow{\$} \{0,1\}^{\mathtt{poly}(\lambda)}$
$(\mathtt{R}, pk^*, \mathbf{c}^*) \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(M, pk, s, c)$	$(\mathtt{R}, pk^*, \mathbf{c}^*) \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(M, pk, s, c)$
return 1 iff $\exists\ (\mathbf{m}^*, \mathbf{r}^*)$ such that	**return** 1 iff $\exists\ (\mathbf{m}^*, \mathbf{r}^*)$ such that
$(\mathbf{c}^* = \mathcal{E}_{pk^*}(\mathbf{m}^*, \mathbf{r}^*)) \wedge$	$(\mathbf{c}^* = \mathcal{E}_{pk^*}(\mathbf{m}^*, \mathbf{r}^*)) \wedge$
$(c \notin \mathbf{c}^* \vee pk \neq pk^*) \wedge$	$(c \notin \mathbf{c}^* \vee pk \neq pk^*) \wedge$
$(\mathbf{m}^* \neq \perp) \wedge$	$(\mathbf{m}^* \neq \perp) \wedge$
$(\mathtt{R}(x, \mathbf{m}^*, pk, pk^*, \mathbf{c}^*) = 1)$	$(\mathtt{R}(\tilde{x}, \mathbf{m}^*, pk, pk^*, \mathbf{c}^*) = 1)$

$$\begin{aligned}
&\textit{if } \text{atk} = \text{cpa} \textit{ then } \mathcal{O}_1(\cdot) = \varepsilon \qquad \textit{and } \mathcal{O}_2(\cdot) = \epsilon,\\
&\textit{if } \text{atk} = \text{cca1} \textit{ then } \mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot) \textit{ and } \mathcal{O}_2(\cdot) = \epsilon,\\
&\textit{if } \text{atk} = \text{cca2} \textit{ then } \mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot) \textit{ and } \mathcal{O}_2(\cdot) = \mathcal{D}^{(c)}_{sk}(\cdot),
\end{aligned}$$

where $\mathcal{D}^{(c)}_{sk}(\cdot)$ stands for a restricted oracle that decrypts any ciphertext but c. One mandates that the distribution M be valid in that $|x| = |x'|$ for any x, x' that have non-zero probability of being sampled. The condition $\mathbf{m}^* \neq \perp$ means that at least one of the components of the vector $\mathbf{c}^*$ must be a valid ciphertext.

The encryption scheme is said NM-ATK secure if for any polynomial-time adversary $\mathcal{A}$, we have $\mathbf{Adv}^{\text{nm-atk}^*}_{\mathcal{A}}(\lambda) \in \mathsf{negl}(\lambda)$.*

As stressed in [15], specific schemes can support invalid public keys (for which no matching private key exists) that look like valid ones. In the above experiments, the adversary might output such an invalid key $pk^\star$ and, in this case, more than one pair (m, r) may correspond to an element of the ciphertext vector $c^\star$. When such an ambiguity arises, both experiments return 1 whenever at least one pair of candidate vectors $(\mathbf{m}^\star, \mathbf{r}^\star)$ happens to satisfy the relation.

In the context of regular non-malleability, simulation and comparison-based definitions were proved equivalent in [4]. In the complete non-malleability setting, the above game-based definition was shown in [26] to imply the simulation-based one in the chosen-plaintext (CPA) scenario. For a fairly wide class of relations (*i.e.*, relations that ignore the input of the challenge public key), a similar implication was established in the CCA1 and CCA2 cases.

2.2 Ingredients and Assumptions

Bilinear Maps. We use groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p and endowed with an efficiently computable map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$, $a, b \in \mathbb{Z}$ and $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$. In this setting, we assume the hardness of this (now classical) problem.

Definition 2. *Let $(\mathbb{G}, \mathbb{G}_T)$ be bilinear groups and $g \in \mathbb{G}$. The* **Decision Bilinear Diffie-Hellman Problem** *(DBDH) is to distinguish the distributions $\{(g^a, g^b, g^c, e(g, g)^{abc}) | a, b, c \xleftarrow{\$} \mathbb{Z}_p^*\}$ and $\{(g^a, g^b, g^c, e(g, g)^z) | a, b, c, z \xleftarrow{\$} \mathbb{Z}_p^*\}$. The advantage of a distinguisher $\mathcal{B}$ is measured by*

$$\mathbf{Adv}^{\mathrm{DBDH}}_{\mathbb{G},\mathbb{G}_T}(\lambda) = \big|\Pr[a, b, c \xleftarrow{\$} \mathbb{Z}_p^* : \mathcal{B}(g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \Pr[a, b, c, z \xleftarrow{\$} \mathbb{Z}_p^* : \mathcal{B}(g^a, g^b, g^c, e(g, g)^z) = 1]\big|.$$

The **Decision Bilinear Diffie-Hellman assumption** *states that, for any probabilistic polynomial time (PPT) algorithm $\mathcal{B}$, $\mathbf{Adv}^{\mathrm{DBDH}}_{\mathbb{G},\mathbb{G}_T}(\lambda) \in \mathsf{negl}(\lambda)$.*

Lossy and All-But-One Trapdoor Functions. Lossy trapdoor functions are collections of functions where injective functions are indistinguishable from many-to-one functions, termed *lossy functions*, that statistically lose all information on the pre-image: each image element has so many equally likely pre-images that even an unbounded adversary cannot determine the right one.

Let $\lambda \in \mathbb{N}$ be a security parameter and let $n, k \in \mathsf{poly}(\lambda)$ be such that $k \leq n$. A collection of (n, k)-lossy trapdoor functions $\Pi_{\mathrm{ltdf}} = (S_{\mathrm{ltdf}}, F_{\mathrm{ltdf}}, F^{-1}_{\mathrm{ltdf}})$ consist of (possibly randomized) efficient algorithms with the following properties.

1. *Easy to sample an injective function with trapdoor:* when $S_{\mathrm{ltdf}}(\lambda, 1)$ outputs (s, t) where s is a function index and t is the trapdoor, $F_{\mathrm{ltdf}}(s, .)$ computes a deterministic injective function over the domain $\{0, 1\}^n$. Besides, $F^{-1}_{\mathrm{ltdf}}(t, .)$ inverts F_{ltdf} in that $F^{-1}_{\mathrm{ltdf}}(t, F_{\mathrm{ltdf}}(s, x)) = x$ for all $x \in \{0, 1\}^n$.
2. *Easy to sample a lossy function:* when $S_{\mathrm{ltdf}}(\lambda, 0)$ outputs $(s, \perp)$, where s is a function index, $F_{\mathrm{ltdf}}(s, .)$ computes a deterministic function over $\{0, 1\}^n$ and with image size 2^{n-k}. The value $r = n - k$ is called *residual leakage*.

3. *Hard to distinguish injective from lossy:* the first outputs of $S_{\mathrm{ltdf}}(\lambda, 1)$ and $S_{\mathrm{ltdf}}(\lambda, 0)$ are computationally indistinguishable. The advantage of a distinguisher is defined analogously to definition 2.

As stressed in [22], the third property implies the infeasibility of inverting an injective function of the collection without knowing the trapdoor.

It was shown in [22] how to efficiently construct such collections in groups where the Decision Diffie-Hellman problem is hard or under worst-case lattice assumptions. Freeman *et al.* [16] gave a more efficient construction (independently suggested in [5]) using the Composite Residuosity assumption [20].

All-but-one (ABO) trapdoor functions are a refinement of the lossy TDF notion. In an ABO collection, each function has several branches, almost all of which are injective and have the same trapdoor. For each function, exactly one branch is lossy and thus has smaller range. The lossy branch is chosen when sampling the function in the collection and remains computationally hidden.

More formally, let $\mathcal{B} = \{B_\lambda\}_{\lambda \in \mathbb{N}}$ be a collection of sets whose element represent the branches. A collection of (n, k)-all-but-one trapdoor functions with branch collections consists of a tuple $\Pi_{\mathrm{abo}} = (S_{\mathrm{abo}}, G_{\mathrm{abo}}, G^{-1}_{\mathrm{abo}})$ of possibly randomized algorithms with the following properties.

1. *Easy to sample a trapdoor function with given lossy branch:* for any $b^* \in B_\lambda$, $S_{\mathrm{abo}}(\lambda, b^*)$ outputs (s, t) where s is a function index and t is the trapdoor. For any $b \in B_\lambda$ such that $b \neq b^*$, $G_{\mathrm{abo}}(s, b, .)$ computes a deterministic injective function over the domain $\{0,1\}^n$. Besides, $G^{-1}_{\mathrm{abo}}(t, b, .)$ inverts it in that $G^{-1}_{\mathrm{abo}}(t, b, G_{\mathrm{abo}}(s, b, x)) = x$ for all $x \in \{0,1\}^n$. Over $\{0,1\}^n$, $G_{\mathrm{abo}}(s, b^*, .)$ computes a deterministic function whose range has size at most 2^{n-k}.
2. *Hidden lossy branch:* for any $b_0^*, b_1^* \in B_\lambda$, the first outputs of $S_{\mathrm{abo}}(\lambda, b_0^*)$ and $S_{\mathrm{abo}}(\lambda, b_1^*)$ are computationally indistinguishable.

3 A Construction Based on the DBDH Assumption

The scheme stems from the Canetti-Halevi-Katz (CHK) methodology [9] that constructs CCA2-secure cryptosystems from weakly secure identity-based encryption schemes. With appropriate modifications, it can be seen as a variant of the CCA2-secure public key encryption scheme derived from the first selective-ID secure IBE system put forth by Boneh and Boyen [6]. The scheme can also be viewed as a CCA2-variant of the escrowed Elgamal encryption scheme of [7][Section 7], where the escrow key is hidden in the CRS and allows dealing with ciphertexts encrypted under new keys in the security proof. For this reason, it appears to be the only known CHK-like system to which the underlying idea applies: the scheme of [19], for instance, is not known to support escrow.

To turn this construction into a NM-CCA2* scheme, we assume that all parties have access to a common reference string comprising the description of bilinear groups $(\mathbb{G}, \mathbb{G}_T)$, random generators in $g, u, v \in \mathbb{G}$ and a strongly unforgeable one-time signature (which are normally part of the public key in the original scheme). Another difference is that the receiver's public key must also be signed

along with other ciphertext components when generating the one-time signature that acts as a "checksum" binding pieces of ciphertexts together.

In the description, we interpret one-time verification keys VK as elements of $\mathbb{Z}_p^*$. Since most such signatures have significantly longer public keys, these should be first hashed onto $\mathbb{Z}_p^*$ using a collision-resistant hash function, the description of which can be included in the reference string.

$\mathcal{CRS}$-$\mathcal{G}(\lambda)$**:** given a security parameter $\lambda \in \mathbb{N}$, choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with random generators $g, u, v \xleftarrow{\$} \mathbb{G}$. Choose also a strongly unforgeable one-time signature scheme $\mathsf{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ (see, e.g., [9] for a definition) The common reference string is $\Sigma = \{\lambda, \mathbb{G}, \mathbb{G}_T, p, g, u, v, \mathsf{Sig}\}$.

$\mathcal{K}(\Sigma)$**:** given Σ, choose $x \xleftarrow{\$} \mathbb{Z}_p^*$ and return the key pair $(pk, sk) = (X = g^x, x)$.

$\mathcal{E}_{pk,\Sigma}(m, r)$**:** given $pk = X$ and Σ, to encrypt $m \in \mathbb{G}_T$ using $r \xleftarrow{\$} \mathbb{Z}_p^*$, generate a one-time signature key pair $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathcal{G}(\lambda)$. Then, compute and return

$$C = \big(\mathsf{VK}, C_1, C_2, C_3, \sigma\big) = \Big(\mathsf{VK},\ g^r,\ (u^{\mathsf{VK}} \cdot v)^r,\ m \cdot e(X, u)^r,\ \sigma\Big)$$

where $\sigma = \mathcal{S}\big(\mathsf{SK}, (C_1, C_2, C_3, pk)\big)$.

$\mathcal{D}_{sk,\Sigma}(C)$**:** given Σ, $sk = x \in \mathbb{Z}_p^*$ and $C = \big(\mathsf{VK}, C_1, C_2, C_3, \sigma\big)$, return $\perp$ if $\mathcal{V}\big(\sigma, \mathsf{VK}, (C_1, C_2, C_3, pk)\big) \neq 1$ or $e(C_1, u^{\mathsf{VK}} \cdot v) \neq e(g, C_2)$. Otherwise, output the plaintext $m = C_3/e(C_1, u)^x$.

Unlike [26], we do not need zero-knowledge proofs vouching for the validity of public keys or ciphertexts: all elements of $\mathbb{G}$ are admissible public keys for which a private key exists and well-formed ciphertexts are also publicly recognizable.

Security. In the event that the adversary does not change the original public key (*i.e.*, $pk^\star = pk$), the proof of NM-CCA2 security implied by [6,9] applies. However, one has to handle ciphertexts encrypted under adversarially-generated public keys. Its strategy to do so, as in the first scheme of [26], is to open such ciphertexts thanks to a trapdoor concealed in the reference string.

Theorem 1. *The scheme is NM-CCA2* assuming the hardness of the DBDH problem and the strong unforgeability of the one-time signature. More precisely, the advantage of any NM-CCA2* adversary $\mathcal{A}$ is bounded by*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{nm-cca2}^*}(\lambda) \leq 2 \cdot \Big(\mathbf{Adv}^{\mathsf{OTS}}(\lambda) + \mathbf{Adv}_{\mathbb{G},\mathbb{G}_T}^{\text{DBDH}}(\lambda)\Big). \tag{1}$$

Proof. The proof consists of a sequence of games where g, u, v are always part of the common reference string Σ. The first game $\text{Game}_0(d)$, where $d \in \{0, 1\}$, operates over the same probability space as the experiment $\mathbf{Expt}^{\text{nm-cca2}^*\text{-d}}$ while the last one is an experiment that should have the same outcome for either value of $d \in \{0, 1\}$. Throughout the sequence, we call $S_i(d)$ the event that the challenger finally outputs 1 in $\text{Game}_i(d)$. We show that any noticeable difference between $\Pr[S_0(0)]$ and $\Pr[S_0(1)]$ (and thus any difference between the outcome of experiments $\mathbf{Expt}^{\text{nm-cca2}^*\text{-d}}$ with $d \in \{0, 1\}$) can be translated into about the same difference between $\Pr[S_5(0)]$ and $\Pr[S_5(1)]$, which is proved to be zero.

Game$_0(d)$: is essentially the experiment $\mathbf{Expt}^{\text{nm-cca2}^*\text{-d}}$. The adversary is provided with a random public key $pk' = X'$ as well as a reference string comprising $u, v \xleftarrow{\$} \mathbb{G}$. Let us define $a = \log_g(X')$ and $b = \log_g(u)$. The adversary $\mathcal{A}$ starts issuing decryption queries. At some point, she makes a challenge query for a plaintext distribution M of her choosing. Then, the challenger picks (m_0, m_1) from M and returns a ciphertext $C' = (\mathsf{VK}', C'_1, C'_2, C'_3, \sigma')$ where $C'_3 = m_d \cdot e(X', u)^{r'}$ for a random $r' = c \in \mathbb{Z}_p^*$ such that $C'_1 = g^c$ and $C'_2 = (u^{\mathsf{VK}'} \cdot v)^c$ and where $\sigma' = \mathcal{S}(\mathsf{SK}', (C'_1, C'_2, C'_3, pk'))$ for a fresh key pair $(\mathsf{VK}', \mathsf{SK}') \leftarrow \mathcal{G}(\lambda)$. Eventually, the adversary outputs a ciphertext-vector $\mathbf{C}^\star$ and a possibly new public key $pk^\star = X^\star$ together with the description of a relation $\mathtt{R}$. At this point, the challenger calls an all powerful oracle that computes $x^\star \in \mathbb{Z}_p^*$ such that $X^\star = g^{x^\star}$ (via presumably exponential-time calculation), which allows decrypting $\mathbf{C}^\star$. The resulting plaintexts are used to evaluate the relation $\mathtt{R}(m_0, \mathbf{m}^\star, \Sigma, pk', pk^\star, \mathbf{C}^\star)$. The challenger returns 1 if the latter holds and 0 otherwise.

Game$_1(d)$: is as Game$_0(d)$ but the one-time key pair $(\mathsf{VK}', \mathsf{SK}')$ is chosen at the beginning of the game. This change is conceptual and $\Pr[S_1(d)] = \Pr[S_0(d)]$.

Game$_2(d)$: we modify the treatment of decryption queries and introduce a rejection rule that also applies to the treatment of $\mathcal{A}$'s output $(\mathbf{C}^\star, pk^\star)$ at the end of the game. Namely, the challenger halts and returns 0 if $\mathcal{A}$ comes up with a *valid* encryption $C = (\mathsf{VK}', C_1, C_2, C_3, \sigma)$ w.r.t. a public key pk such that either

- the query occurs before $\mathcal{A}$ receives the challenge ciphertext C'.
- C appears after the challenge C' but $(C_1, C_2, C_3, pk, \sigma) \neq (C'_1, C'_2, C'_3, pk', \sigma')$.

Game$_2(d)$ and Game$_1(d)$ proceed identically unless this event, that we call F_2, occurs. In the first case, $\mathcal{A}$ was necessarily able to forge a signature σ without seeing a single signature (or even the verification key VK'). The second case implies a polynomial (strong) signature forger: indeed, even if pk is a new public key, the challenger detects F_2 and halts before having to invoke its super-polynomial oracle. Therefore, $|\Pr[S_2(d)] - \Pr[S_1(d)]| \leq \Pr[F_2] \leq \mathbf{Adv}^{\mathsf{OTS}}(\lambda)$. In particular, the above implies that, if $\mathcal{A}$'s output $(\mathbf{C}^\star, pk^\star)$ involves a new public key $pk^\star \neq pk'$, the signature verification key VK' cannot appear within $\mathbf{C}^\star$ unless $\mathcal{A}$ was able to defeat the one-time signature security.

Game$_3(d)$: we change the generation of the common reference string Σ. At the outset of the game, the challenger chooses $\beta \xleftarrow{\$} \mathbb{Z}_p^*$ and defines $v = u^{-\mathsf{VK}'} \cdot g^\beta$. In the challenge phase, the triple (C'_1, C'_2, C'_3) is generated as

$$C'_1 = g^{r'} = g^c, \qquad C'_2 = (g^{r'})^\beta = (g^c)^\beta \qquad C'_3 = m_d \cdot e(g^a, g^b)^c.$$

These modifications do not alter the adversary's view. In particular, the distributions of Σ and C' remain the same and we have $\Pr[S_3(d)] = \Pr[S_2(d)]$.

Game$_4(d)$: we change the treatment of all ciphertexts produced by $\mathcal{A}$ (those queried for decryption as well as the elements of $\mathbf{C}^\star$). For a public key pk (which

is either X' or a possibly new public key $X^\star$ chosen by the adversary), such a ciphertext $C = (\mathsf{VK}, C_1, C_2, C_3, \sigma)$ is declared invalid if σ does not properly verify under VK or if $e(C_1, u^{\mathsf{VK}} \cdot v) \neq e(g, C_2)$. Now, let us assume that C is valid. We necessarily have $\mathsf{VK} \neq \mathsf{VK}'$ since the rejection rule of $\mathrm{Game}_2(d)$ would apply otherwise (recall that, even if $pk^\star = pk'$ in $\mathcal{A}$'s final output, $\mathbf{C}^\star$ cannot contain copies of the challenge ciphertext C'). Given that $C_1 = g^r$ and $C_2 = (u^{\mathsf{VK}} \cdot v)^r$ for some unknown $r \in \mathbb{Z}_p^*$, the challenger can compute $u^r = (C_2/C_1^\beta)^{1/(\mathsf{VK}-\mathsf{VK}')}$ which in turn reveals $e(X^\star, u^r)$ and the plaintext $m = C_3/e(X^\star, u)^r$.

The decryption oracle is perfectly simulated and the treatment of $(\mathbf{C}^\star, pk^\star)$ is the same as if it were made by decrypting $\mathbf{C}^\star$ using the private key associated with $pk^\star = X^\star$. Therefore, we have $\Pr[S_4(d)] = \Pr[S_3(d)]$. In this game, exponents $a, b, c \in \mathbb{Z}_p^*$ are not explicitly handled any longer by the challenger that only manipulates them via g^a, g^b, g^c and $e(g,g)^{abc}$.

Game$_5(d)$: we let $\mathrm{Game}_5(d)$ be as $\mathrm{Game}_4(d)$ except that the pairing value $e(X', u)^{r'} = e(g,g)^{abc}$, that was used to compute $C_3' = m_d \cdot e(X', u)^{r'}$, is now replaced by a random element $\gamma \xleftarrow{\$} \mathbb{G}_T$. We have a transition based on indistinguishability and can thus write $|\Pr[S_5(d)] - \Pr[S_4(d)]| \leq \mathbf{Adv}^{\mathrm{DBDH}}_{\mathbb{G},\mathbb{G}_T}(\lambda)$.

When gathering probabilities, we find the inequality

$$|\Pr[S_0(d)] - \Pr[S_5(d)]| \leq \mathbf{Adv}^{\mathsf{OTS}}(\lambda) + \mathbf{Adv}^{\mathrm{DBDH}}_{\mathbb{G},\mathbb{G}_T}(\lambda)$$

for $d \in \{0,1\}$. Now, we claim that $\Pr[S_5(1)] = \Pr[S_5(0)]$. In $\mathrm{Game}_5(d)$, C_3' is a random element of $\mathbb{G}_T$ that perfectly hides m_d. Even an unbounded adversary's view is identically distributed for either value of $d \in \{0,1\}$. Given that

$$\begin{aligned}\mathbf{Adv}^{\text{nm-cca2}^*}_{\mathcal{A}}(\lambda) = |\Pr[S_0(1)] - \Pr[S_0(0)]| &\leq |\Pr[S_0(1)] - \Pr[S_5(1)]| \\ &+ |\Pr[S_5(1)] - \Pr[S_5(0)]| + |\Pr[S_5(0)] - \Pr[S_0(0)]|,\end{aligned}$$

when combining the above, we find the claimed upper bound (1). □

4 A Construction Based on Lossy Trapdoor Functions

This construction makes use of lossy and all-but-one (ABO) functions and is a natural modification of a cryptosystem described in [22]. The latter has the important property of being *witness recovering* (which is a unique feature for a standard model public key encryption scheme). The fact that receivers retrieve senders' random coins upon decryption is precisely what allows proving complete non-malleability when positioning oneself in the CRS model.

The main difference w.r.t. [22] is that, instead of being chosen by the user upon key generation, the family of all-but-one functions is defined as part of the reference string and thus cannot be changed by the adversary. As in section 3, in order to prevent malleability attempts on the public key, the latter is part of the "message" that is one-time-signed in the ciphertext generation.

$\mathcal{CRS}$-$\mathcal{G}(\lambda)$**:** given a security parameter $\lambda \in \mathbb{N}$,

1. Define the message space as $\{0,1\}^\ell$ for some $\ell \in \mathbb{N}$.
2. Generate a collection $\Pi_{\text{abo}} = (S_{\text{abo}}, G_{\text{abo}}, G_{\text{abo}}^{-1})$ of (n, k')-ABO trapdoor functions having branches $B_\lambda = \{0,1\}^v$, for some $v \in \mathsf{poly}(\lambda)$.
3. Choose a collection $\Pi_{\text{ltdf}} = (S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ of (n, k)-lossy TDFs. It is required that $k + k' \geq n + \kappa$, where $\kappa = \kappa(n) = \omega(\log n)$ [1] is such that $\kappa > \ell + \log(1/\varepsilon)$ for some $\varepsilon \in \mathsf{negl}(\lambda)$.
4. Select an ABO function with random lossy branch: set $b_0 \xleftarrow{\$} \{0,1\}^v$, $(s_0, t_0) \leftarrow S_{\text{abo}}(\lambda, b_0)$ and discard the trapdoor t_0 as well as b_0.
5. Choose a strong one-time signature scheme $\mathsf{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ and a pairwise independent (see, e.g., [22]) hash function $h : \{0,1\}^n \to \{0,1\}^\ell$.
6. Output $\Sigma = \{\lambda, \ell, \Pi_{\text{abo}}, \Pi_{\text{ltdf}}, s_0, \mathsf{Sig}, h\}$.

$\mathcal{K}(\Sigma)$**:** given Σ, generate an injective trapdoor function $(s,t) \leftarrow S_{\text{ltdf}}(\lambda, 1)$ from the lossy TDF family and set $(pk, sk) = (s, t)$.

$\mathcal{E}_{pk,\Sigma}(m, r)$**:** to encrypt m under $pk = s$ using the randomness $r \in \{0,1\}^n$, generate a one-time signature key pair $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathcal{G}(\lambda)$ and compute

$$C = (\mathsf{VK}, C_1, C_2, C_3, \sigma) = \Big(\mathsf{VK},\ F_{\text{ltdf}}(s, r),\ G_{\text{abo}}(s_0, \mathsf{VK}, r),\ m \oplus h(r),\ \sigma\Big)$$

where $\sigma = \mathcal{S}\big(\mathsf{SK}, (C_1, C_2, C_3, pk)\big)$.

$\mathcal{D}_{sk,\Sigma}(C)$**:** given $C = (\mathsf{VK}, C_1, C_2, C_3, \sigma)$, return $\perp$ if $\mathcal{V}\big(\sigma, \mathsf{VK}, (C_1, C_2, C_3, pk)\big) \neq 1$. Otherwise, compute $r = F_{\text{ltdf}}^{-1}(t, C_1)$ and return $m = C_3 \oplus h(r)$ if $C_2 = G_{\text{abo}}(s_0, \mathsf{VK}, r)$ and $C_1 = F_{\text{ltdf}}(s, r)$. Otherwise, return $\perp$.

The security proof is naturally based on [22] and is detailed in the full paper.

In comparison with the first scheme, this one has less compact ciphertexts with currently known instantiations of lossy TDFs. The DDH-based one [22] provides ciphertexts comprising $O(n)$ elements (or $O(n/\log(n))$ with space optimizations) of the DDH-hard group and the size of ciphertexts is thus super-linear in the security parameter λ. The instantiation based on the Composite Residuosity assumption [16] yields ciphertexts and public keys that have linear size in the security parameter but remain significantly larger than in section 3.

On the other hand, the lossy-TDF-based construction gives instantiations based on long-lived (non-pairing-related) assumptions.

5 Conclusion

We described the first efficient completely non-malleable public key encryption schemes in the common reference string model. Both schemes are proved to meet the game-based definition of Ventre and Visconti. The basic idea employed was to move to a common parameter string rather than employing the inefficient NIZK. In this scenario, we identified schemes that implement the notion, the simplicity of the transformation of the basic schemes to our systems assures they are quite practical. One related open question is the exact connection of the game-based definition and Fischlin's simulation-based one.

[1] That is, $\kappa(n)$ grows faster than $c \cdot \log n$ for any constant $c > 0$.

References

1. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, p. 26. Springer, Heidelberg (1998)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS (1993)
3. Bellare, M., Rogaway, P.: Optimal asymmetric encryption - how to encrypt with RSA. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
4. Bellare, M., Sahai, A.: Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, p. 519. Springer, Heidelberg (1999)
5. Boldyreva, A., Fehr, S., O'Neill, A.: On Notions of Security for Deterministic Encryption, and Efficient Constructions without Random Oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)
6. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
7. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. SIAM Journal of Computing 32(3), 586–615 (2003)
8. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. Journal of the ACM 51(4) (2004)
9. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
10. Choi, S.-G., Dachman-Soled, D., Malkin, T., Wee, H.: Black-box construction of a non-malleable encryption scheme from any semantically secure one. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 427–444. Springer, Heidelberg (2008)
11. Cramer, R., Shoup, V.: A Practical Public-Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, p. 13. Springer, Heidelberg (1998)
12. Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, p. 45. Springer, Heidelberg (2002)
13. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: STOC 1991, pp. 542–552. ACM Press, New York (1991)
14. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. on Information Theory 29(2), 198–207 (1983)
15. Fischlin, M.: Completely Non-malleable Schemes. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 779–790. Springer, Heidelberg (2005)
16. Freeman, D., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More Constructions of Lossy and Correlation-Secure Trapdoor Functions. In: PKC 2010. LNCS. Springer, Heidelberg (2010)
17. Goldwasser, S., Micali, S.: Probabilistic Encryption. J. Comput. Syst. Sci. 28(2) (1984)
18. Herzog, J., Liskov, M., Micali, S.: Plaintext Awareness via Key Registration. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 548–564. Springer, Heidelberg (2003)

19. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
20. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, p. 223. Springer, Heidelberg (1999)
21. Pass, R., Shelat, A., Vaikuntanathan, V.: Construction of a Non-malleable Encryption Scheme from Any Semantically Secure One. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 271–289. Springer, Heidelberg (2006)
22. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC 2008. ACM Press, New York (2008)
23. Rackoff, C., Simon, D.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
24. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC 1990. ACM Press, New York (1990)
25. Sahai, A.: Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In: FOCS 1999 (1999)
26. Ventre, C., Visconti, I.: Completely non-malleable encryption revisited. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 65–84. Springer, Heidelberg (2008)
27. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)

Polynomial-Space Approximation of No-Signaling Provers

Tsuyoshi Ito

Institute for Quantum Computing and School of Computer Science
University of Waterloo
200 University Ave. W., Waterloo, ON N2L 3G1, Canada
tsuyoshi@iqc.ca

Abstract. In two-prover one-round interactive proof systems, no-signaling provers are those who are allowed to use arbitrary strategies, not limited to local operations, as long as their strategies cannot be used for communication between them. The study of multi-prover interactive proof systems with no-signaling provers has been motivated by the study of those with provers sharing quantum states. The relation between them is that no-signaling strategies include all the strategies realizable by provers sharing arbitrary entangled quantum states, and more. It was known that PSPACE $\subseteq$ MIP$^{\mathrm{ns}}(2,1) \subseteq$ EXP, where MIP$^{\mathrm{ns}}(2,1)$ is the class of languages having a two-prover one-round interactive proof system with no-signaling provers.

This paper shows MIP$^{\mathrm{ns}}(2,1)$ = PSPACE. This is proved by constructing a fast parallel algorithm which approximates within an additive error the maximum winning probability of no-signaling players in a given cooperative two-player one-round game. The algorithm uses the fast parallel algorithm for the mixed packing and covering problem by Young (FOCS 2001).

1 Introduction

1.1 Background

Nonlocality [3] is a peculiar property of quantum mechanics and has applications to quantum information processing. Following Cleve, Høyer, Toner and Watrous [7], quantum nonlocality can be naturally expressed in terms of a cooperative two-player one-round game with imperfect information, which is a game played by two players and a referee as follows. The players are kept in separate rooms so that they cannot communicate with each other. The referee chooses a pair of questions according to some probability distribution, and sends one question to each player. Each player replies with an answer to the referee, and the referee declares whether the two players jointly win or jointly lose according to the questions and the answers. The players know the protocol used by the referee including the probability distribution of the pair of questions and how the referee determines the final outcome of the game, but none of the players knows the question sent to the other player. The aim of the players is to win

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 140–151, 2010.

the game with as high probability as possible, and the maximum winning probability is called the *value* of the game. In this framework, a Bell inequality is an inequality stating an upper bound on the value of a game of this kind when provers are not allowed to perform any quantum operations, and the violation of a Bell inequality means that the game value increases when provers are allowed to share a quantum state before the game starts.

The complexity of finding or approximating the value of a game has been one of the most fundamental problems in computational complexity theory. The computational model based on cooperative multi-player games is called multi-prover interactive proof systems and was introduced by Ben-Or, Goldwasser, Kilian and Wigderson [4] for a cryptographic purpose.[1] It turned out that this computational model is extremely powerful: multi-prover interactive proof systems exactly characterize NEXP [1,11], even in the most restricted settings with two provers, one round and an exponentially small one-sided error [10]. Scaling down this result implies that, given the description of a cooperative game, approximating the best strategy even in a very weak sense is NP-hard.

Cleve, Høyer, Toner and Watrous [7] connected computational complexity theory and quantum nonlocality and raised the question on the complexity of approximating the value of a cooperative game with imperfect information in the case where the players are allowed to share quantum states or, in terms of interactive proof systems, the computational power of multi-prover interactive proof systems with entangled provers. Kobayashi and Matsumoto [23] considered another quantum variation of multi-prover interactive proof systems where the verifier can also use quantum information and can exchange quantum messages with provers, which is a multi-prover analogue of quantum interactive proof systems [29]. In Ref. [23], it was shown that allowing the provers to share at most polynomially many qubits does not increase the power of multi-prover interactive proof systems beyond NEXP (even if the verifier is quantum). Although studied intensively, the power of multi-prover interactive proof systems with provers allowed to share arbitrary quantum states has been still largely unknown.

The notion of no-signaling strategies was first studied in physics in the context of Bell inequalities by Khalfin and Tsirelson [22] and Rastall [27], and it has gained much attention after being reintroduced by Popescu and Rohrlich [25]. The acceptance probability of the optimal no-signaling provers is often useful as an upper bound on the acceptance probability of entangled provers because no-signaling strategies have a simple mathematical characterization.

Kempe, Kobayashi, Matsumoto, Toner and Vidick [21] prove, among other results, that every language in PSPACE has a two-prover one-round interactive proof system which has one-sided error $1-1/\mathrm{poly}$ even if honest provers are unentangled and dishonest provers are allowed to have prior entanglement of any size (the proof is in Ref. [20]). Ito, Kobayashi and Matsumoto [15] improve their result to an exponentially small one-sided error by considering no-signaling provers; more specifically, they prove that the soundness of the protocol in Ref. [21] actually holds against arbitrary no-signaling provers, then use the parallel repetition

[1] Because of this connection, we use "player" and "prover" synonymously in this paper.

theorem for no-signaling provers [14]. We note that the soundness analysis of Ref. [15] is somewhat simpler than that of Ref. [21].

Repeating the protocol of Ref. [21] in parallel as is done in Ref. [15] results in the protocol identical to the one used by Cai, Condon and Lipton [6] to prove that every language in PSPACE has a two-prover one-round interactive proof system with an exponentially small one-sided error in the classical world. Therefore, an implication of Ref. [15] is that the protocol in Ref. [6] has an unexpected strong soundness property: the protocol remains to have an exponentially small error even if we allow the two provers to behave arbitrarily as long as they are no-signaling.

Given that the soundness of protocols can be perhaps analyzed easier against no-signaling provers than against entangled provers, it is natural to ask whether the result of Ref. [15] can be extended to a class larger than PSPACE. This raises a question to characterize the class $\mathrm{MIP}^{\mathrm{ns}}(2,1)$ of languages having a two-prover one-round interactive proof system with no-signaling provers with bounded two-sided error. The abovementioned result in Ref. [15] implies $\mathrm{MIP}^{\mathrm{ns}}(2,1) \supseteq \mathrm{PSPACE}$. On the other hand, Preda [26] pointed out $\mathrm{MIP}^{\mathrm{ns}}(2,1) \subseteq \mathrm{EXP}$.

1.2 Our Results

Our main result is:

Theorem 1. $\mathrm{MIP}^{\mathrm{ns}}(2,1) \subseteq \mathrm{PSPACE}$.

An immediate corollary obtained by combining Theorem 1 with the abovementioned result in Ref. [15] is the following exact characterization of the class $\mathrm{MIP}^{\mathrm{ns}}(2,1)$:

Corollary 1. $\mathrm{MIP}^{\mathrm{ns}}(2,1) = \mathrm{PSPACE}$*, and this is achievable with exponentially small one-sided error, even if honest provers are restricted to be unentangled.*

This puts the proof system of Ref. [6] in a rather special position: while other two-prover one-round interactive proof systems [1,9,10] work with the whole NEXP, the one in Ref. [6] attains the best achievable by two-prover one-round interactive proof systems with bounded two-sided error that are sound against no-signaling provers, and at the same time, it achieves an exponentially small one-sided error.

At a lower level, our result is actually a parallel algorithm to approximately decide[2] the value of a two-player one-round game for no-signaling players as follows. For a two-player one-round game G, $w_{\mathrm{ns}}(G)$ is the value of G for no-signaling provers and $|G|$ is the size of G, both of which will be defined in Section 2.1.

Theorem 2. *There exists a parallel algorithm which, given a two-player one-round game G and numbers $0 \leq s < c \leq 1$ such that either $w_{\mathrm{ns}}(G) \leq s$ or*

[2] The algorithm stated in Theorem 2 can be converted to an algorithm to approximate $w_{\mathrm{ns}}(G)$ within an additive error in a standard way. See Remark 2 in Section 3.

$w_{\mathrm{ns}}(G) \geq c$, decides which is the case. The algorithm runs in parallel time polynomial in $\log|G|$ and $1/(c-s)$ and total work polynomial in $|G|$ and $1/(c-s)$.

Theorem 1 follows in a standard manner from the polynomial equivalence between parallel time and sequential space complexity [5] by applying the algorithm of Theorem 2 to the exponential-size game naturally arising from a two-prover one-round interactive proof system. This approach is similar to that of the recent striking result on the PSPACE upper bound on QIP [17] as well as other complexity classes related to quantum interactive proof systems, i.e. QRG(1) [19] and QIP(2) [18].[3] The detail of the derivation of Theorem 1 from Theorem 2 is omitted due to space limitation.

The construction of the parallel algorithm in Theorem 2 is much simpler than those used in Refs. [17,18,19] because our task can be formulated as solving a *linear* program of a certain special form approximately instead of a *semidefinite* program. This allows us to use the fast parallel algorithm for the mixed packing and covering problem by Young [32].[4]

1.3 Organization of the Paper

The rest of this paper is organized as follows. Section 2 gives the definitions used later and states the result by Young [32] about a fast parallel approximation algorithm for the mixed packing and covering problem. Section 3 states a technical lemma (Lemma 1) which represents the no-signaling value of a game by the optimal value of a linear program closely related to the mixed packing and covering problem, and proves Theorem 2 assuming this lemma by using Young's fast parallel algorithm. Section 4 proves Lemma 1. Section 5 concludes the paper by discussing some natural open problems and their difficulties.

2 Preliminaries

We assume the familiarity with the notion of multi-prover interactive proof systems. Readers are referred to the textbook by Goldreich [12].

2.1 Games

A protocol of a two-prover one-round interactive proof system defines a game of exponential size for each instance. Here we give a formal definition of *games*.

[3] Do not be confused by an unfortunate inconsistency as for whether the number in the parenthesis represents the number of *rounds* or *turns*, where one round consists of two turns. The "1" in QRG(1) and the "2" in QIP(2) represent the number of turns whereas the "1" in $\mathrm{MIP}^{\mathrm{ns}}(2,1)$ represents the number of rounds just in the same way as the "1" in $\mathrm{MIP}(2,1)$.

[4] Alternatively, it is possible to prove Theorem 2 by using the multiplicative weights update method, a common key technique used in Refs. [17,18,19]. See Remark 1 in Section 3 for a brief explanation.

A *two-prover one-round game*, or simply a *game* in this paper, is played by two cooperative provers called prover 1 and prover 2 with help of a verifier who enforces the rule of the game. A game is formulated as $G = (Q_1, Q_2, A_1, A_2, \pi, R)$ by nonempty finite sets Q_1, Q_2, A_1 and A_2, a probability distribution π over $Q_1 \times Q_2$, and a function $R\colon Q_1 \times Q_2 \times A_1 \times A_2 \to [0,1]$. As is customary, we write $R(q_1, q_2, a_1, a_2)$ as $R(a_1, a_2 \mid q_1, q_2)$.

In this game, the verifier generates a pair of questions $(q_1, q_2) \in Q_1 \times Q_2$ according to the probability distribution π, and sends q_1 to the prover 1 and q_2 to the prover 2. Each prover i ($i \in \{1,2\}$) sends an answer $a_i \in A_i$ to the verifier without knowing the question sent to the other prover. Finally, the verifier accepts with probability $R(a_1, a_2 \mid q_1, q_2)$ and rejects with probability $1 - R(a_1, a_2 \mid q_1, q_2)$. The provers try to make the verifier accept with as high probability as possible.

The *size* $|G|$ of the game G is defined as $|G| = |Q_1||Q_2||A_1||A_2|$.

A *strategy* in a two-prover one-round game G is a family $p = (p_{q_1 q_2})$ of probability distributions on $A_1 \times A_2$ indexed by $(q_1, q_2) \in Q_1 \times Q_2$. As is customary, the probability $p_{q_1 q_2}(a_1, a_2)$ is written as $p(a_1, a_2 \mid q_1, q_2)$. A strategy p is said to be *no-signaling* if it satisfies the following *no-signaling conditions*:

- The marginal probability $p_1(a_1 \mid q_1) = \sum_{a_2} p(a_1, a_2 \mid q_1, q_2)$ does not depend on q_2.
- Similarly, $p_2(a_2 \mid q_2) = \sum_{a_1} p(a_1, a_2 \mid q_1, q_2)$ does not depend on q_1.

The *acceptance probability* of a strategy p is given by

$$\sum_{q_1 \in Q_1, q_2 \in Q_2} \pi(q_1, q_2) \sum_{a_1 \in A_1, a_2 \in A_2} R(a_1, a_2 \mid q_1, q_2) p(a_1, a_2 \mid q_1, q_2).$$

The *no-signaling value* $w_{\mathrm{ns}}(G)$ of G is the maximum acceptance probability over all no-signaling strategies.

2.2 Interactive Proof Systems

In a *two-prover one-round interactive proof system*, a verifier is a randomized polynomial-time process. He is given an input string x, produces questions to the two provers, and decides whether he accepts or rejects the answers from the provers. A two-prover one-round interactive proof system defines a game $G^{(x)}$ in a natural way for each input string x.

Let $c, s\colon \mathbb{Z}_{\geq 0} \to [0,1]$ be functions such that $c(n) > s(n)$ for every n. The two-prover one-round interactive proof system is said to *recognize a language*[5] L *with completeness acceptance probability at least* $c(n)$ *and soundness error at most* $s(n)$ *with no-signaling provers* when the following conditions are satisfied.

[5] Although we define $\mathrm{MIP}^{\mathrm{ns}}(2,1)$ as a class of languages in this paper to keep the notations simple, we could alternatively define $\mathrm{MIP}^{\mathrm{ns}}(2,1)$ as the class of *promise problems* [8] (see also Section 2.4.1 of Ref. [12]) recognized by a two-prover one-round interactive proof system with no-signaling provers. A generalization of Theorem 1 to the case of promise problems is straightforward.

Completeness $x \in L \implies w_{\mathrm{ns}}(G^{(x)}) \geq c(|x|)$.
Soundness $x \notin L \implies w_{\mathrm{ns}}(G^{(x)}) \leq s(|x|)$.

In particular, the proof system is said to *recognize L with bounded error with no-signaling provers* if $1/(c(n) - s(n))$ is bounded by a polynomial in n and the binary representations of $c(n)$ and $s(n)$ are computable in time polynomial in n. We denote by $\mathrm{MIP}^{\mathrm{ns}}(2,1)$ the class of languages L which are recognized by a two-prover one-round interactive proof system with bounded error with no-signaling provers.

2.3 Mixed Packing and Covering Problem

The *mixed packing and covering problem* is the linear feasibility problem to find a vector $x \in \mathbb{R}^N$ satisfying $Ax \leq b$, $Cx \geq d$ and $x \geq 0$, where matrices A, C and vectors b, d are given and the entries of A, b, C, d are all nonnegative. The constraints of the form $Ax \leq b$ are called *packing constraints*, and the constraints of the form $Cx \geq d$ are *covering constraints*. For $r \geq 1$, an *r-approximate solution* is a vector $x \geq 0$ such that $Ax \leq rb$ and $Cx \geq d$.

Theorem 3 (Young [32]). *There exists a parallel algorithm which, given an instance (A, b, C, d) of the mixed packing and covering problem and a number $\varepsilon > 0$, either claims that the given instance does not have a feasible solution, or finds a $(1+\varepsilon)$-approximate solution. If the size of A and C are $M_1 \times N$ and $M_2 \times N$, respectively, then the algorithm runs in parallel time polynomial in $\log M_1$, $\log M_2$, $\log N$ and $1/\varepsilon$ and total work polynomial in M_1, M_2, N and $1/\varepsilon$.*

3 Proof of Theorem 2

This section states a technical lemma and gives a proof of Theorem 2 assuming this lemma.

Let $G = (Q_1, Q_2, A_1, A_2, \pi, R)$ be a game. Let $\pi_1(q_1) = \sum_{q_2 \in Q_2} \pi(q_1, q_2)$ and $\pi_2(q_2) = \sum_{q_1 \in Q_1} \pi(q_1, q_2)$ be the marginal distributions.

Lemma 1. *The no-signaling value $w_{\mathrm{ns}}(G)$ is equal to the optimal value of the following linear program (1). Here $x_1(q_1)$ for $q_1 \in Q_1$, $x_2(q_2)$ for $q_2 \in Q_2$, $y_1(q_1, q_2, a_1)$ for $(q_1, q_2, a_1) \in Q_1 \times Q_2 \times A_1$ and $y_2(q_1, q_2, a_2)$ for $(q_1, q_2, a_2) \in Q_1 \times Q_2 \times A_2$ are variables.*

$$\text{Minimize} \quad \sum_{q_1} x_1(q_1) + \sum_{q_2} x_2(q_2), \tag{1a}$$

$$\text{Subject to} \quad y_1(q_1, q_2, a_1) + y_2(q_1, q_2, a_2) \leq \pi(q_1, q_2)\big(2 - R(a_1, a_2 \mid q_1, q_2)\big), \quad \forall q_1, q_2, a_1, a_2, \tag{1b}$$

$$x_1(q_1) + \sum_{q_2} y_1(q_1, q_2, a_1) \geq \pi_1(q_1), \qquad \forall q_1, a_1, \tag{1c}$$

$$x_2(q_2) + \sum_{q_1} y_2(q_1, q_2, a_2) \geq \pi_2(q_2), \qquad \forall q_2, a_2, \tag{1d}$$

$$y_1(q_1, q_2, a_1) \le \pi(q_1, q_2), \qquad \forall q_1, q_2, a_1, \tag{1e}$$
$$y_2(q_1, q_2, a_2) \le \pi(q_1, q_2), \qquad \forall q_1, q_2, a_2, \tag{1f}$$
$$x_1(q_1) \ge 0, \qquad \forall q_1, \tag{1g}$$
$$x_2(q_2) \ge 0, \qquad \forall q_2, \tag{1h}$$
$$y_1(q_1, q_2, a_1) \ge 0, \qquad \forall q_1, q_2, a_1, \tag{1i}$$
$$y_2(q_1, q_2, a_2) \ge 0, \qquad \forall q_1, q_2, a_2. \tag{1j}$$

Note that in the the linear program (1), each variable is constrained to be nonnegative, each constraint is either a packing constraint or a covering constraint, and the objective function is monotone. We defer a proof of Lemma 1 to Section 4.

Lemma 2. *Let $G = (Q_1, Q_2, A_1, A_2, \pi, R)$ be a game and $0 \le s < c \le 1$. Consider the instance of the mixed packing and covering problem consisting of a constraint $\sum_{q_1} x_1(q_1) + \sum_{q_2} x_2(q_2) \le s$ and the constraints (1b)–(1j). Let $\varepsilon = (c - s)/4$. Then,*

(i) If $w_{\mathrm{ns}}(G) \le s$, this instance has a feasible solution.
(ii) If $w_{\mathrm{ns}}(G) \ge c$, this instance does not have a $(1 + \varepsilon)$-approximate solution.

Proof. (i) Clear from Lemma 1.

(ii) We prove the contrapositive. Assume that $(\tilde{x}_1, \tilde{x}_2, \tilde{y}_1, \tilde{y}_2)$ is a $(1 + \varepsilon)$-approximate solution of this mixed packing and covering problem, and let $x_1(q_1) = \tilde{x}_1(q_1) + \varepsilon\pi_1(q_1)$, $x_2(q_2) = \tilde{x}_2(q_2) + \varepsilon\pi_2(q_2)$, $y_1(q_1, q_2, a_1) = \tilde{y}_1(q_1, q_2, a_1)/(1 + \varepsilon)$ and $y_2(q_1, q_2, a_2) = \tilde{y}_2(q_1, q_2, a_2)/(1 + \varepsilon)$. It is easy to verify that (x_1, x_2, y_1, y_2) is a feasible solution of the linear program (1), and the objective value of this solution is $\sum_{q_1} x_1(q_1) + \sum_{q_2} x_2(q_2) = \sum_{q_1} \tilde{x}_1(q_1) + \varepsilon \sum_{q_1} \pi_1(q_1) + \sum_{q_2} \tilde{x}_2(q_2) + \varepsilon \sum_{q_2} \pi_2(q_2) \le (1 + \varepsilon)s + 2\varepsilon < s + 3\varepsilon < c$. Therefore, the optimal value of the linear program (1) is less than c, which implies $w_{\mathrm{ns}}(G) < c$ by Lemma 1. □

Proof (of Theorem 2). Apply Theorem 3 to the instance of the mixed packing and covering problem in Lemma 2. □

Remark 1. It is easy to see that adding the constraints $x_1(q_1) \le \pi_1(q_1)$ for $q_1 \in Q_1$ and $x_2(q_2) \le \pi_2(q_2)$ for $q_2 \in Q_2$ to the instance of the mixed packing and covering problem in Lemma 2 does not change the feasibility or approximate feasibility. The resulting linear program has a constant "width" in the sense stated in Theorem 2.12 of Plotkin, Shmoys and Tardos [24] with a suitable tolerance vector. See Ref. [24] for relevant definitions. This gives an alternative proof of Theorem 2 which uses the algorithm of Ref. [24] instead of the algorithm of Ref. [32]. With this modification, it is also possible to use the multiplicative weights update method instead of the algorithm of Ref. [24], which explains the similarity to the method used in Refs. [17,18,19].

Remark 2. Given Theorem 2, it is easy to approximate $w_{\mathrm{ns}}(G)$ within additive error ε (rather than deciding whether $w_{\mathrm{ns}}(G) \le s$ or $w_{\mathrm{ns}}(G) \ge c$) in parallel time polynomial in $\log|G|$ and $1/\varepsilon$ and total work polynomial in $|G|$ and $1/\varepsilon$. This can be done by trying all the possibilities of $s = k\varepsilon$ and $c = (k + 1)\varepsilon$ for integers k in the range $0 \le k \le 1/\varepsilon$ in parallel, or by using the binary search.

4 Formulating No-Signaling Value by a Linear Program with Packing and Covering Constraints

This section proves Lemma 1.

Our starting point is the following linear program (2), whose optimal value is equal to the no-signaling value $w_{\mathrm{ns}}(G)$ of G by definition:

$$\text{Maximize} \quad \sum_{q_1,q_2} \pi(q_1,q_2) \sum_{a_1,a_2} R(a_1,a_2 \mid q_1,q_2) p(a_1,a_2 \mid q_1,q_2), \tag{2a}$$

$$\text{Subject to} \quad \sum_{a_2} p(a_1,a_2 \mid q_1,q_2) = p_1(a_1 \mid q_1), \qquad \forall q_1,q_2,a_1, \tag{2b}$$

$$\sum_{a_1} p(a_1,a_2 \mid q_1,q_2) = p_2(a_2 \mid q_2), \qquad \forall q_1,q_2,a_2, \tag{2c}$$

$$\sum_{a_1,a_2} p(a_1,a_2 \mid q_1,q_2) = 1, \qquad \forall q_1,q_2, \tag{2d}$$

$$p(a_1,a_2 \mid q_1,q_2) \geq 0, \qquad \forall q_1,q_2,a_1,a_2. \tag{2e}$$

We transform this linear program (2) as follows. First, we replace the constraint (2d) by two constraints $\sum_{a_1} p_1(a_1 \mid q_1) = 1$ for all q_1 and $\sum_{a_2} p_2(a_2 \mid q_2) = 1$ for all q_2. Next, we add redundant constraints $p_1(a_1 \mid q_1) \geq 0$ for all q_1 and a_1 and $p_2(a_2 \mid q_2) \geq 0$ for all q_2 and a_2. Next, we relax the constraints (2b) and (2c) to inequalities. This results in the following linear program (3):

$$\text{Maximize} \quad \sum_{q_1,q_2} \pi(q_1,q_2) \sum_{a_1,a_2} R(a_1,a_2 \mid q_1,q_2) p(a_1,a_2 \mid q_1,q_2), \tag{3a}$$

$$\text{Subject to} \quad \sum_{a_2} p(a_1,a_2 \mid q_1,q_2) \leq p_1(a_1 \mid q_1), \qquad \forall q_1,q_2,a_1, \tag{3b}$$

$$\sum_{a_1} p(a_1,a_2 \mid q_1,q_2) \leq p_2(a_2 \mid q_2), \qquad \forall q_1,q_2,a_2, \tag{3c}$$

$$\sum_{a_1} p_1(a_1 \mid q_1) = 1, \qquad \forall q_1, \tag{3d}$$

$$\sum_{a_2} p_2(a_2 \mid q_2) = 1, \qquad \forall q_2, \tag{3e}$$

$$p(a_1,a_2 \mid q_1,q_2) \geq 0, \qquad \forall q_1,q_2,a_1,a_2, \tag{3f}$$

$$p_1(a_1 \mid q_1) \geq 0, \qquad \forall q_1,a_1, \tag{3g}$$

$$p_2(a_2 \mid q_2) \geq 0, \qquad \forall q_2,a_2. \tag{3h}$$

Claim 1. *The optimal values of the linear programs (2) and (3) are equal.*

Proof. Let w and w' be the optimal values of the linear programs (2) and (3), respectively. Since we only added redundant constraints and relaxed some of the constraints, $w \leq w'$ is obvious. To prove $w \geq w'$, let $(\tilde{p}, p_1, p_2)$ be a feasible solution of (3). We will construct p such that (p, p_1, p_2) is a feasible solution

of the linear program (2) and $p(a_1, a_2 \mid q_1, q_2) \geq \tilde{p}(a_1, a_2 \mid q_1, q_2)$ for every q_1, q_2, a_1, a_2.

Fix any $q_1 \in Q_1$ and $q_2 \in Q_2$. Let $s_{q_1q_2}(a_1) = p_1(a_1 \mid q_1) - \sum_{a_2 \in A_2} \tilde{p}(a_1, a_2 \mid q_1, q_2)$ for each a_1, $t_{q_1q_2}(a_2) = p_2(a_2 \mid q_2) - \sum_{a_1 \in A_1} \tilde{p}(a_1, a_2 \mid q_1, q_2)$ for each a_2 and $u_{q_1q_2} = 1 - \sum_{a_1 \in A_1, a_2 \in A_2} \tilde{p}(a_1, a_2 \mid q_1, q_2)$. Since $(\tilde{p}, p_1, p_2)$ satisfies the constraints (3b) and (3c), we have $s_{q_1q_2}(a_1) \geq 0$ and $t_{q_1q_2}(a_2) \geq 0$. Eqs. (3d) and (3e) implies that $\sum_{a_1 \in A_1} s_{q_1q_2}(a_1) = \sum_{a_2 \in A_2} t_{q_1q_2}(a_2) = u_{q_1q_2}$.

We define $p(a_1, a_2 \mid q_1, q_2)$ by

$$p(a_1, a_2 \mid q_1, q_2) = \begin{cases} \tilde{p}(a_1, a_2 \mid q_1, q_2) + \dfrac{s_{q_1q_2}(a_1) t_{q_1q_2}(a_2)}{u_{q_1q_2}}, & \text{if } u_{q_1q_2} > 0, \\ \tilde{p}(a_1, a_2 \mid q_1, q_2), & \text{if } u_{q_1q_2} = 0. \end{cases}$$

Then $p(a_1, a_2 \mid q_1, q_2) \geq \tilde{p}(a_1, a_2 \mid q_1, q_2)$ for every q_1, q_2, a_1, a_2. The constraints (2b)–(2e) are easy to verify. □

By the duality theorem of linear programming (see e.g. Section 7.4 of Ref. [28]), the linear program (3) has the same optimal value as the following linear program (4):

$$\text{Minimize} \quad \sum_{q_1} x_1(q_1) + \sum_{q_2} x_2(q_2), \tag{4a}$$

$$\text{Subject to} \quad z_1(q_1, q_2, a_1) + z_2(q_1, q_2, a_2) \geq \pi(q_1, q_2) R(a_1, a_2 \mid q_1, q_2), \quad \forall q_1, q_2, a_1, a_2, \tag{4b}$$

$$x_1(q_1) \geq \sum_{q_2} z_1(q_1, q_2, a_1), \quad \forall q_1, a_1, \tag{4c}$$

$$x_2(q_2) \geq \sum_{q_1} z_2(q_1, q_2, a_2), \quad \forall q_2, a_2, \tag{4d}$$

$$z_1(q_1, q_2, a_1) \geq 0, \quad \forall q_1, q_2, a_1, \tag{4e}$$

$$z_2(q_1, q_2, a_2) \geq 0, \quad \forall q_1, q_2, a_2. \tag{4f}$$

Note that the constraints (4c)–(4f) imply $x_1(q_1) \geq 0$ and $x_2(q_2) \geq 0$, and therefore adding these redundant nonnegativity constraints to the linear program (4) does not change its optimal value.

Let (x_1, x_2, z_1, z_2) be a feasible solution of the linear program (4). If $z_1(q_1, q_2, a_1) > \pi(q_1, q_2)$ for some q_1, q_2, a_1, we can replace $z_1(q_1, q_2, a_1)$ by $\pi(q_1, q_2)$ without violating any constraints or increasing the objective value. The same holds for $z_2(q_1, q_2, a_2)$. Therefore, adding the constraints $z_1(q_1, q_2, a_1) \leq \pi(q_1, q_2)$ for all q_1, q_2, a_1 and $z_2(q_1, q_2, a_2) \leq \pi(q_1, q_2)$ for all q_1, q_2, a_2 does not change the optimal value.

Replacing the variables $z_1(q_1, q_2, a_1)$ by $\pi(q_1, q_2) - y_1(q_1, q_2, a_1)$ and $z_2(q_1, q_2, a_2)$ by $\pi(q_1, q_2) - y_2(q_1, q_2, a_2)$, we obtain Lemma 1.

5 Concluding Remarks

This paper gave the exact characterization of the simplest case of multi-prover interactive proof systems with no-signaling provers: $\mathrm{MIP}^{\mathrm{ns}}(2, 1) = \mathrm{PSPACE}$.

A natural direction seems to be to extend this result to show a PSPACE upper bound on a class containing $\mathrm{MIP}^{\mathrm{ns}}(2,1)$. Below we discuss some hurdles in doing so.

More than two provers. In the completely classical case, a many-prover one-round interactive proof system can be transformed to a two-prover one-round interactive proof system by using the oracularization technique, and therefore $\mathrm{MIP}(\mathrm{poly},1) \subseteq \mathrm{MIP}(2,1)$. The same transformation is not known to preserve soundness in the case of no-signaling provers even when the original proof system uses three provers.[6] As a result, whether or not $\mathrm{MIP}^{\mathrm{ns}}(3,1) \subseteq \mathrm{MIP}^{\mathrm{ns}}(2,1)$ is unknown, and our result does not imply $\mathrm{MIP}^{\mathrm{ns}}(3,1) \subseteq \mathrm{PSPACE}$.

To extend the current proof to $\mathrm{MIP}^{\mathrm{ns}}(3,1)$, the main obstacle is to extend Claim 1, which replaces equations by inequalities. Somewhat surprisingly, it does not seem that an analogous claim can be proved for three provers by a straightforward extension of the current proof of Claim 1.

More than one round. The proof of Claim 1 seems to work in the case of two-prover systems with polynomially many rounds. However, in a linear program corresponding to (4), an upper bound on the variables z_1 and z_2 becomes exponentially large and the current proof does not work even in the case of two-prover two-round systems with adaptive questions or two-prover $\omega(\log n)$-round systems with non-adaptive questions.

Quantum verifier and quantum messages. The notion of no-signaling strategies can be extended to the case of quantum messages [2,13] (Ref. [2] uses the term "causal" instead of "no-signaling"). This allows us to define e.g. the class $\mathrm{QMIP}^{\mathrm{ns}}(2,2)$ of languages having a *quantum* two-prover one-round (two-turn) interactive proof system with no-signaling provers. The class $\mathrm{QMIP}^{\mathrm{ns}}(2,2)$ contains both $\mathrm{MIP}^{\mathrm{ns}}(2,1)$ and $\mathrm{QIP}(2)$, and it would be nice if the method of Ref. [18] and ours can be unified to give $\mathrm{QMIP}^{\mathrm{ns}}(2,2) = \mathrm{PSPACE}$. One obvious obstacle is how to extend the fast parallel algorithm in Ref. [18] for the special case of semidefinite programming to the case of $\mathrm{QMIP}^{\mathrm{ns}}(2,2)$. Another obstacle is again Claim 1: the current proof of Claim 1 essentially constructs a joint probability distribution over (q_1, q_2, a_1, a_2) from its marginal distributions over (q_1, q_2, a_1) and (q_1, q_2, a_2), and this kind of *state extension* is not always possible in the quantum case [30,31].

Acknowledgments. The author thanks Rahul Jain, Julia Kempe, Hirotada Kobayashi, Sarvagya Upadhyay and John Watrous for helpful discussions. He also thanks an anonymous reviewer of the QIP 2010 workshop for helpful comments and for pointing out a small omission in the proof of Lemma 1 in an earlier version of this paper, and an anonymous reviewer of the ICALP 2010 conference for helpful comments. He is also grateful to NSERC, CIFAR and QuantumWorks for support.

[6] The Magic Square game in Ref. [7] is a counterexample which shows that this transformation cannot be used alone to reduce the number of provers from three to two in the case of *entangled* provers because it sometimes transforms a three-prover game whose entangled value is less than 1 to a two-prover game whose entangled value is equal to 1 [16]. The situation might be different in the case of no-signaling provers.

References

1. Babai, L., Fortnow, L., Lund, C.: Non-deterministic exponential time has two-prover interactive protocols. Computational Complexity 1(1), 3–40 (1991)
2. Beckman, D., Gottesman, D., Nielsen, M.A., Preskill, J.: Causal and localizable quantum operations. Physical Review A 64(052309) (2001)
3. Bell, J.S.: On the Einstein-Podolsky-Rosen paradox. Physics 1, 195–200 (1964)
4. Ben-Or, M., Goldwasser, S., Kilian, J., Wigderson, A.: Multi-prover interactive proofs: How to remove intractability assumptions. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC), pp. 113–131 (1988)
5. Borodin, A.: On relating time and space to size and depth. SIAM Journal on Computing 6(4), 733–744 (1977)
6. Cai, J.Y., Condon, A., Lipton, R.J.: PSPACE is provable by two provers in one round. Journal of Computer and System Sciences 48(1), 183–193 (1994)
7. Cleve, R., Høyer, P., Toner, B., Watrous, J.: Consequences and limits of nonlocal strategies. In: Proceedings: Nineteenth Annual IEEE Conference on Computational Complexity (CCC), pp. 236–249 (2004)
8. Even, S., Selman, A.L., Yacobi, Y.: The complexity of promise problems with applications to public-key cryptography. Information and Control 61(2), 159–173 (1984)
9. Feige, U.: On the success probability of the two provers in one-round proof systems. In: Proceedings of the Sixth Annual Structure in Complexity Theory Conference (SCT), pp. 116–123 (1991)
10. Feige, U., Lovász, L.: Two-prover one-round proof systems: Their power and their problems. In: Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing (STOC), pp. 733–744 (1992)
11. Fortnow, L., Rompel, J., Sipser, M.: On the power of multi-prover interactive protocols. Theoretical Computer Science 134(2), 545–557 (1994)
12. Goldreich, O.: Computational Complexity: A Conceptual Perspective. Cambridge University Press, Cambridge (2008)
13. Gutoski, G.: Properties of local quantum operations with shared entanglement. Quantum Information and Computation 9(9–10), 739–764 (2009)
14. Holenstein, T.: Parallel repetition: Simplifications and the no-signaling case. Theory of Computing 5(Article 8), 141–172 (2009)
15. Ito, T., Kobayashi, H., Matsumoto, K.: Oracularization and two-prover one-round interactive proofs against nonlocal strategies. In: Proceedings: Twenty-Fourth Annual IEEE Conference on Compuational Complexity (CCC), pp. 217–228 (2009)
16. Ito, T., Kobayashi, H., Preda, D., Sun, X., Yao, A.C.C.: Generalized Tsirelson inequalities, commuting-operator provers, and multi-prover interactive proof systems. In: Proceedings: Twenty-Third Annual IEEE Conference on Compuational Complexity (CCC), pp. 187–198 (2008)
17. Jain, R., Ji, Z., Upadhyay, S., Watrous, J.: QIP=PSPACE. In: Proceedings of the Forty-Second Annual ACM Symposium on Theory of Computing STOC (2010) (to appear) arXiv:0907.4737v2 [quant-ph]
18. Jain, R., Upadhyay, S., Watrous, J.: Two-message quantum interactive proofs are in PSPACE. In: Proceedings: Fiftieth Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 534–543 (2009)
19. Jain, R., Watrous, J.: Parallel approximation of non-interactive zero-sum quantum games. In: Proceedings: Twenty-Fourth Annual IEEE Conference on Compuational Complexity (CCC), pp. 243–253 (2009)

20. Kempe, J., Kobayashi, H., Matsumoto, K., Toner, B., Vidick, T.: Entangled games are hard to approximate. arXiv:0704.2903v2 [quant-ph] (2007)
21. Kempe, J., Kobayashi, H., Matsumoto, K., Toner, B., Vidick, T.: Entangled games are hard to approximate. In: Proceedings: Forty-Ninth Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 447–456 (2008)
22. Khalfin, L.A., Tsirelson, B.S.: Quantum and quasi-classical analogs of Bell inequalities. In: Symposium on the Foundations of Modern Physics, pp. 441–460 (1985)
23. Kobayashi, H., Matsumoto, K.: Quantum multi-prover interactive proof systems with limited prior entanglement. Journal of Computer and System Sciences 66(3), 429–450 (2003)
24. Plotkin, S.A., Shmoys, D.B., Tardos, É.: Fast approximation algorithms for fractional packing and covering problems. Mathematics of Operations Research 20(2), 257–301 (1995)
25. Popescu, S., Rohrlich, D.: Quantum nonlocality as an axiom. Foundations of Physics 24(3), 379–385 (1994)
26. Preda, D.: Private communication
27. Rastall, P.: Locality, Bell's theorem, and quantum mechanics. Foundations of Physics 15(9), 963–972 (1985)
28. Schrijver, A.: Theory of Linear and Integer Programming. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, Chichester (1986)
29. Watrous, J.: PSPACE has constant-round quantum interactive proof systems. Theoretical Computer Science 292(3), 575–588 (2003)
30. Werner, R.F.: An application of Bell's inequatilies to a quantum state extension problem. Letters in Mathematical Physics 14(4), 359–363 (1989)
31. Werner, R.F.: Remarks on a quantum state extension problem. Letters in Mathematical Physics 19(4), 319–326 (1990)
32. Young, N.E.: Sequential and parallel algorithms for mixed packing and covering. In: Proceedings: Forty-Second IEEE Symposium on Foundations of Computer Science (FOCS), pp. 538–546 (2001)

From Secrecy to Soundness: Efficient Verification via Secure Computation (Extended Abstract)

Benny Applebaum[1,⋆], Yuval Ishai[2,⋆⋆], and Eyal Kushilevitz[3,⋆⋆⋆]

[1] Computer Science Department, Weizmann Institute of Science
[2] Computer Science Department, Technion and UCLA
[3] Computer Science Department, Technion

Abstract. We study the problem of *verifiable computation* (VC) in which a computationally weak client wishes to delegate the computation of a function f on an input x to a computationally strong but untrusted server. We present new general approaches for constructing VC protocols, as well as solving the related problems of program checking and self-correcting. The new approaches reduce the task of verifiable computation to suitable variants of secure multiparty computation (MPC) protocols. In particular, we show how to efficiently convert the secrecy property of MPC protocols into soundness of a VC protocol via the use of a message authentication code (MAC). The new connections allow us to apply results from the area of MPC towards simplifying, unifying, and improving over previous results on VC and related problems.

In particular, we obtain the following concrete applications: (1) The first VC protocols for arithmetic computations which only make a black-box use of the underlying field or ring; (2) a non-interactive VC protocol for boolean circuits in the preprocessing model, conceptually simplifying and improving the online complexity of a recent protocol of Gennaro et al. (Cryptology ePrint Archive: Report 2009/547); (3) $\mathbf{NC^0}$ self-correctors for complete languages in the complexity class $\mathbf{NC^1}$ and various log-space classes, strengthening previous $\mathbf{AC^0}$ correctors of Goldwasser et al. (STOC 2008).

1 Introduction

In the *verifiable computation* (VC) problem, we have a computationally weak device (client) who wishes to compute a complex function f on an input x. The client is too weak to compute f on its own and so it delegates the computation to a computationally strong server. However, the client does not trust the server

⋆ Work done in part while visiting Princeton University. Supported by Koshland Fellowship, and NSF grants CNS-0627526, CCF-0426582 and CCF-0832797.

⋆⋆ Supported by BSF grant 2008411, ISF grant 1310/06, and NSF grants 0830803, 0716835, 0627781.

⋆⋆⋆ Work done in part while visiting UCLA. Supported by BSF grant 2008411 and ISF grant 1310/06.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 152–163, 2010.

and therefore would like to be able to verify the correctness of the computation without investing too much resources. One may also consider a stronger variant of the problem in which, in addition to the ability to detect arbitrary errors, the client should be able to *correct* the errors as long as the server is somewhat correct with respect to some predefined distribution over the inputs. This corresponds to the scenario where the server, or alternatively a program locally run by the client, makes "unintentional" errors on some fraction of the inputs (e.g., due to implementation bugs). Still, a malicious server should not be able to fool the client to accept an erroneous answer. We refer to this variant as the *correctable verifiable computation* (CVC) problem.

VC and CVC are fundamental problems which were extensively studied in various settings, originating from the early works on interactive proofs [4,21] and program checking [7,9,28]. Recent advances in technology further motivate these problems. On the one hand, computationally weak peripheral devices such as smart phones and netbooks are becoming increasingly common; on the other hand, the increasing use of distributed computing over the internet also makes strong servers more commonly available. Indeed, the growing volume of outsourcable computation in the form of "cloud computing" or in projects like SETI@Home has attracted a renewed interest in the VC problem, and a considerable amount of research was devoted to these problems in the last few years [22,18,19,20,15,11]. See [20,15] for further discussion of applications as well as a survey of related work.

In this work, we present new general approaches for solving the VC and CVC problems, as well as the related problems of program checking and self-correcting. Our approaches employ variants of secure multi-party computation (MPC), converting their *secrecy* features into *soundness* by making additional use of basic cryptographic primitives such as message authentication codes (MACs) and symmetric encryption. By instantiating these general approaches we obtain several improvements over previous results in this area. We stress that the idea of employing secrecy in the context of verification is not unique to our work. This idea can be traced back to the first works on interactive proofs and program checking [4,21,7,28] and is also implicit in more recent works in the area [18,19,15,11]. Our work provides *new* approaches for converting secrecy into soundness that have advantages of generality, efficiency, and simplicity over previous approaches.

1.1 Background

Before introducing our new approaches to VC, we review some of the relevant notions and previous approaches.

MPC and related primitives. A protocol for secure two-party computation [32,17] allows two parties, each holding a private input x_i, to compute a function on their joint input without revealing any additional information to each other. That is, the first (resp., second) party learns the output of some predefined function $f_1(x_1, x_2)$ (resp., $f_2(x_1, x_2)$) without learning any additional information about

x_2 (resp., x_1). Unless otherwise mentioned, we only require *computational* security against *semi-honest* parties who operate as instructed by the protocol (but may try to learn additional information from the messages they observe), and make no secrecy or correctness requirements in the presence of malicious parties.

We will be interested in secure protocols in which one of the parties is restricted in its computational resources in a way that prevents it from computing the output on its own, even when given the entire input. Such restrictions may include bounds on sequential or parallel time (either with or without preprocessing), on space complexity, on arithmetic circuit complexity, etc. We will refer to the weak party as the *client* and to the strong party as the *server*. In contrast to the typical study of feasibility questions in the area of secure computation, in the context of restricted clients it makes sense to consider even functions for which only the client holds an input, as well as protocols for such functions with perfect or statistical rather than computational security. (The existence of statistically secure two-party protocols can be ruled out for almost all natural functions which depend on both inputs.)

A client-server protocol in which only the client has an input and gets an output is called an *instance-hiding* (IH) protocol [1,5]. For simplicity, we will mainly restrict the attention to *one-round* (or two-message) IH protocols which consist of a single "query" from the client to the server followed by a single "answer" from the server to the client. A natural extension to multi-round IH protocols is deferred to the full version.

Central to this work is a different (and in some sense more stringent) variant of client-server protocols, in which only the client has an input x but *both parties* learn the same output $f(x)$. One-round protocols of this type coincide with the notion of *randomized encoding* from [23,3]. A randomized encoding (RE) of f is a function $\hat{f}(x;r)$ whose output on a uniformly random and secret r can be used to decode $f(x)$ but reveals no additional information about x. In the corresponding client-server protocol, the client picks r at random and sends the encoded output $\hat{f}(x;r)$ as a query to the server; the server decodes the output $f(x)$ and sends it back as an answer to the client. In the full version, we discuss applications of an interactive variant of this primitive, referred to as *interactive randomized encoding* (IRE).[1] RE and IRE protocols can be easily converted into IH protocols with the same number of rounds by modifying the function f to compute an *encryption* of the output under a secret key selected by the client. A similar transformation in the other direction seems unlikely. As a notable example, the existence of a *fully homomorphic encryption* scheme [16] implies a one-round IH protocol for any polynomial-time computable f in which the client's time complexity grows only linearly with the input length, whereas the existence of similar RE protocols is an open problem.

Note that for all of the above types of client-server protocols, we are not concerned with protecting the privacy of the server, since the server has no input. We can therefore assume, without loss of generality, that an honest server is deterministic.

[1] This generalization is somewhat subtle in that it involves a nontrivial security requirement against a malicious server; see full version for details.

The traditional approach for VC. The literature on program checking and interactive proofs already makes an implicit use of a general transformation from IH to VC.[2] For simplicity, we restrict the attention to one-round IH protocols. The basic idea is roughly as follows. The client uses the IH protocol to compute $f(x)$ while hiding the input x from the server, except that it randomly mixes the "real" IH query with an appropriately-distributed random query whose correct answer is somehow known (more on this below). The client accepts the output obtained from the real IH instance only if the server's answer on the dummy query is identical to the precomputed answer. By the hiding property, the server cannot distinguish the real query from the dummy one, and so a cheating server will be caught with probability $\frac{1}{2}$. (The soundness can be amplified via repetition.) More formally, this approach requires two building blocks: (1) a one-round IH protocol, in which the client efficiently maps x to a query $\hat{x}$ such that, given the server's answer $g(\hat{x})$ (together with the client's randomness), it is possible to efficiently recover $f(x)$; and (2) a *solved instance generator* (SIG): an efficient way for generating a random instance r for g (under the distribution defined by the client's query in the IH scheme) together with its solution $g(r)$.

We summarize the advantages and disadvantages of the SIG+IH approach. On the positive side, IH is a relatively liberal notion which is implied by secure computation in the semi-honest model, and SIG is easy in many cases, e.g., it is given "for free" if polynomial-time preprocessing is allowed before *each* real query. (See [11] for a the usefulness of this approach when applied with IH based on fully homomorphic encryption, and [15,11] for the further use of fully homomorphic encryption towards *reusable* preprocessing.) On the negative side, SIGs are not *always* easy to construct (for instance, the absence of parallel SIGs significantly complicated the parallel checkers of [19] and prevented [19] from achieving highly parallel correctors for, say, log-space complexity classes). Another disadvantage of the SIG+IH approach has to do with the overhead of soundness amplification: in order to achieve soundness error of $2^{-\tau}$, the VC protocol needs to invoke the IH and SIG protocols $\Omega(\tau)$ times.

1.2 Our Solutions

Motivated by the above disadvantages of the traditional approach, we present two new approaches for transforming variants of MPC into VC or CVC.

Construction 1: VC from RE+MAC. Our first approach is based on a novel combination of an RE (or IRE protocol) with a private-key signature scheme (also known as message authentication code or MAC). Unlike previous approaches, we employ secrecy in order to hide the MAC's secret key, rather than the inputs of the computation. The idea is as follows: Given an input x, the client asks the server to compute $y = f(x)$ and, in addition, to generate a signature on $f(x)$ under a private key k which is chosen randomly by the client. The latter request is computed via an RE protocol that hides the private key from the server. More

[2] The following formulation is similar to the one from Section 1.2 of [18]; see [9,14,19,11] for other variants and applications of this approach.

precisely, the client who holds both x and k, invokes an RE such that both parties learn the function $g(x,k) = \mathtt{MAC}_k(f(x))$. The client then accepts the answer y if and only if the result of the protocol is a valid signature on y under the key k. The soundness of the protocol follows by showing that a cheating server, which fools the client to accept an erroneous $y^* \neq f(x)$, can be used to either break the privacy of the RE or to forge a valid signature on a new message. For this argument to hold, it is crucial for the RE to be secure in the following sense: a malicious server should not be able to force an erroneous output which violates privacy; that is, one should be able to simulate erroneous outputs solely based on the correct outputs. In the case of RE (where there are only two messages), this requirement follows automatically from the basic secrecy requirement against a *semi-honest* server. In the interactive setting, we show that such useful IRE protocols can be extracted from various MPC protocols that appear in the literature.

Note that the above approach eliminates both of the disadvantages of the SIG+IH approach mentioned above, at the expense of replacing IH with the stronger RE primitive and (slightly) increasing the complexity of the function f by applying a MAC computation to its output.

Construction 2: CVC from RE + One-time pad. The previous construction does not seem to apply to the case of CVC. Our second construction yields a CVC protocol and, as can be expected, is somewhat less efficient. The starting point is the well-known CVC version of the SIG+IH approach [9]. In this version, dummy IH queries obtained via SIG are mixed with (randomized) IH queries for the real instance. The client first verifies that most of the dummy queries were answered correctly, and then outputs the majority vote of the outputs obtained from the real answers. Our main goal here is to eliminate the need for SIG primitive. The idea is to generate the dummy queries by applying the IH to some default input x_0 whose image $y_0 = f(x_0)$ is known, and compare the *outputs* obtained from these dummy queries to the known output y_0. (The fixed value of y_0 can be "wired" into the description of the client and used in all subsequent invocations.) By the hiding property the messages of the client are distributed according to some fixed universal probability distribution D which does not depend on the actual input. By using standard concentration bounds, one can show that the client will correct the errors of a "buggy" (rather than malicious) server which doesn't err too much over messages drawn from D. Intuitively, the privacy of the IH protocol also prevents a *malicious* server from cheating, as such a server cannot distinguish between a "dummy" query to a "real" one, and therefore a cheating behavior will be be detected (whp). However, this intuition is inaccurate as, in general, even if the server cannot distinguish dummy queries from real ones, it might be able to apply the same strategy to all the queries such that errors will be generated only in the real queries.[3] Fortunately, this can be fixed by requiring

[3] Consider, for example, an IH in which a client whose input equals to the all zero string, ignores the server's answers and outputs $f(\mathbf{0})$. A CVC protocol which makes use of such an IH together with $x_0 = \mathbf{0}$ can be trivially broken by a malicious server which sends erroneous answers.

an additional *sensitivity* property: any erroneous message of the server should lead to an erroneous answer of the client. To achieve this property, we combine an RE protocol with a one-time pad encryption scheme. That is, we employ an RE for the function $g(x,k) = k \oplus f(x)$ where k is used as a one-time pad. The use of one-time pad transforms the RE to a "sensitive" IH.

Compared to the traditional approach, the above approach eliminates the need for SIG at the expense of strengthening the IH primitive.

2 Applications

By instantiating our generic approaches, we derive new constructions of VC and CVC protocols in several settings.

2.1 Online/Offline Non-Interactive VC

In the online/offline setting [20,15], the client can afford to invest a lot of computational resources in a preprocessing phase before seeing the actual input x, but only a small amount of computational resources after x is known. (Imagine a smart card which is initialized in a secure environment and later operates in a hostile environment with an untrusted server.) Since communication may also be limited, especially for weak devices, we would like the protocol to be non-interactive. That is, in the offline phase the client should perform some (possibly expensive) computation and send the result (the "public-key") to the server or publish it somewhere.[4] In the online phase, when the client obtains its input x, it should send a single computationally-cheap message to the server. The server then computes the result without any intermediate interaction with the client, which in the meantime can be disconnected from the network. At the end of the computation, the server publishes an answer. Based on this answer, the client recovers the result $y = f(x)$ or announces an error in the case of a cheating server.

We would like to minimize the client's online time complexity ideally to be only linear in the input and output length of f. We also require the complexity of the server to be polynomial in the time complexity of f. There are only few known solutions that yield almost optimal non-interactive VCs (NIVCs) for general Boolean functions. These include the constructions of Micali [30] in the random oracle model, the construction of Goldwasser et al. and Kalai and Raz [20,26] for low-depth circuits, and the recent construction by Gennaro et al. [15] for polynomial-size Boolean circuits which relies on the existence of one-way functions.

[4] In a concurrent and independent work, Chung, Kalai, and Vadhan [11] obtain a qualitatively stronger type of non-interactive VC protocols, where the offline preprocessing phase can only involve a *local* computation performed by the client with no additional interaction. The applications we present only apply to the weaker model of non-interactive VC, but obtain better online efficiency in this model.

While these constructions provide good solutions for binary computations, they suffer from large overhead in the case of arithmetic computations. Indeed, a client who wishes to delegate a computational task which should be performed over non-binary domains such as the integers, finite-precision reals, matrices, or elements of a big finite ring, has no choice but to translate the computation into a binary circuit and then apply one of the above solutions. This results in large computational and communication overhead which heavily depends on the exact structure of the underlying ring.[5] A much more satisfactory solution would be to describe the computation in an arithmetic model in which computational operations are performed over some ring $\mathcal{R}$ and then employ an arithmetic NIVC. More formally, we would like to have a protocol in which both the server and the client only have a *black-box* access to $\mathcal{R}$. This black-box access enables the client and server to perform ring operations and sample random ring elements, but the correspondence between ring elements and their identifiers (or even the exact size of the ring) will be unknown to the algorithms. The black-box ring model allows to abstract away the exact structure of the underlying ring, and thus to obtain protocols in which the number of ring operations does not depend on the actual algebraic structure of $\mathcal{R}$. Unfortunately, all the above constructions do not seem to achieve such a result. The reason is that the main tools employed by these constructions (i.e., PCP machinery in the case of [30,20], and Yao's garbled circuit [32] in the case of [15]) do not seem to work in the arithmetic black-box model, even for the special case of black-box fields.

Our results. We obtain NIVCs in the black-box ring model for arithmetic branching programs [6] (ABPs) which are the arithmetic analog of log-space counting classes.[6]

Theorem 1 (informal). *Assuming the existence of one-way functions, there exists a NIVC in the BBR model with perfect completeness and computational soundness error* $\mathrm{neg}(\tau)$ *where* τ *is the security parameter. The complexity of the offline phase and the server's complexity are* $\mathrm{poly}(s, \tau)$*, the time complexity of the online phase is* $O(n\tau)$ *at the query step and* $O(\tau)$ *at the verification step, where* n *is the input length, and* s *is the size of the ABP.*

To the best of our knowledge, this is the first construction of VC in the black-box arithmetic model, even for the case of black-vox fields and even if many rounds of interaction are allowed. The main ingredient is a new construction of arithmetic REs with low online complexity (which is based on [24,12]). The NIVC is obtained by plugging this RE (together with black-box arithmetic MAC) into our RE+MAC approach.

Optimized and simplified NIVC for Boolean circuits. As an additional application, we simplify the recent online/offline NIVC of Gennaro, Genty and Parno [15]

[5] For example, even in the case of finite fields with n-bit elements, the size of the best known Boolean multiplication circuits is $\omega(n \log n)$; the situation is significantly worse for other useful rings, such as matrix rings.

[6] Such programs are quite expressive and are capable of emulating arithmetic formulas.

as well as improve its online efficiency. Specifically, GGP constructed a NIVC for every polynomial-size circuit $f : \{0,1\}^n \to \{0,1\}^m$ with low online complexity as well as low amortized offline complexity. This is achieved in two steps. First, a basic NIVC with low online complexity is constructed by relying on special properties of Yao's garbled circuit (GC) construction and, then, a fully homomorphic encryption scheme is used to reduce the amortized complexity of the offline phase. Our version of the basic protocol follows immediately by instantiating the RE+MAC approach with computationally-sound RE based on GC [2]. This leads to the following theorem:

Theorem 2 (informal). *Assuming the existence of one-way functions, every function $f : \{0,1\}^n \to \{0,1\}^m$ of circuit size s, can be realized by a NIVC with perfect completeness, computational soundness error of* $\mathrm{neg}(\tau) + 2^{-\sigma}$ *(where τ and σ are computational and statistical security parameters, respectively), and complexity as follows.* ***Client:*** *Offline complexity $O(s \cdot \tau + \sigma)$, and online complexity of $O(n \cdot \tau)$ at the query step, and $O(m + \sigma)$ at the verification step.* ***Server:*** *complexity of $O(s \cdot \tau + \sigma)$.*

This theorem simplifies and slightly improves the efficiency of the basic GGP scheme. Simplification comes from the fact that we do not need to rely on any specific properties of Yao's encoding other than its standard security and the well known ability to break the computation of the GC into an offline phase and a cheap online phase. Moreover, we also get an efficiency advantage: in the online phase of the GGP protocol the client needs to get an encryption key for each bit of the output. Hence, both the communication and computation complexity at the verification stage are $O(m\tau)$ where τ is a computational security parameter. In our case, the client needs to get (in addition to the output) only a short certification string of size σ, where σ is a *statistical* security parameter, and so the complexity is $O(m + \sigma)$. This difference can be significant for computations in which the output is much longer than the input (and shorter than the circuit size). For instance, think of image processing algorithms which return "enhanced" versions of low-quality pictures or movies.[7] Finally, we observe that the second step of the [15] construction in which the offline complexity is being amortized forms a general transformation and so it can be used to amortize the offline stage of our construction as well. (A similar observation was made independently and concurrently by [11].)

Program checking and correcting. In the setting of program checking [7,9], one would like to have a VC protocol for a function f in which the power of the honest server is limited: it can only compute the function f itself. That is, the honest server always responds to a message q by the message $f(q)$. Such a VC

[7] One thing to note, though, is that if the client already knows a candidate y for $f(x)$ (obtained either from the server or from some other source) then the GGP approach can be applied for the boolean function $g(x, y)$ which verifies that $y = f(x)$. In such a case, the communication to the client will only be $m + O(\tau)$, but the online communication to the server grows asymptotically when y is longer than x.

protocol is called *program self-checker*.[8] Indeed, a checker can be used to check the correctness of a possibly faulty program for f on a given input, by letting the program play the role of the server. Similarly, a CVC in which the server can be implemented by f is called a self-tester/corrector pair, as it allows to test whether a given program is not too faulty, and if so to correct it.

Minimizing the parallel complexity. Rubinfeld [31] initiated the study of the parallel complexity (circuit depth) of program checkers and correctors, and showed that some non-trivial functions can be checked by $\mathbf{AC^0}$ checkers (i.e., constant depth circuits with AND and OR gates of unbounded fan-in). Goldwasser et al. [19] proved several surprising results about the parallel complexity of program checking and correcting. Among other things, they showed that a rich family of combinatorial and algebraic languages, namely, all the complete languages in the complexity classes $\mathbf{NC^1}, \oplus\mathbf{L}/poly, \mathbf{Mod_kL}/poly$, can be checked in $\mathbf{NC^0}$ (i.e., by constant depth circuits with bounded-fan in gates) and corrected in $\mathbf{AC^0}$.[9] We improve this result by showing that all these languages can be also *corrected* in $\mathbf{NC^0}$:

Theorem 3 (informal). *Every language which is complete for one of the complexity classes* $\mathbf{NC^1}, \oplus\mathbf{L}/poly, \mathbf{Mod_kL}/poly$ *under* $\mathbf{NC^0}$ *Karp reductions can be checked, tested and corrected by an* $\mathbf{NC^0}$ *client with perfect completeness and (arbitrarily small) constant statistical soundness error. Correction succeeds with arbitrary large constant probability (say* $2/3$*) as long as the server's error probability is bounded away from* $1/2$ *(e.g.,* $1/3$*).*

Furthermore, our corrector (and checker) only makes a constant number of calls to the program in a *non-adaptive* way. This is contrasted with the constructions of [19] which make an adaptive use of the program even in the case of checkers. (This difference seems to be inherent to the "composition approach" of [19] which breaks the computation to subroutines and checks them by sub-checkers.) As a concrete example of our improvement, consider the function Det which computes the determinant of an $n \times n$ matrix over a field $\mathbb{F}_p$ of fixed prime order. Since Det is complete for the class $\mathbf{Mod_pL}/poly$ [29], we can get an $\mathbf{NC^0}$ tester/correcter for the determinant over any fixed finite field which makes a constant number of calls to the program. Previous correctors either had polynomial depth [9], or were implemented in $\mathbf{AC^0}$ and made large (non-constant) number of calls to the program [19]. (See [19, Table 1]). Our constructions are obtained by instantiating the RE+OTP approach with the $\mathbf{NC^0}$ REs of [3].

Additional properties. We mention that most of our protocols satisfy additional useful properties. For example, we can add input-privacy and allow the client (or checker) employ the program without revealing its input. In some cases, we

[8] In fact, the notion defined here is slightly stronger than the original definition of [7], and corresponds to *adaptive checkers* as in [8].

[9] Recall that there is a considerable gap between these two classes, as in $\mathbf{NC^0}$ circuits each bit of the output depends only on a constant number of input bits; thus, an $\mathbf{NC^0}$ circuit cannot compute even an n-bit AND gate.

can also add a form of zero-knowledge property: the client learns only the value $f(x)$ and no other additional information that she cannot compute by herself using her own *weak* resources. This may be useful when the server is getting paid for his work and does not want to be abused and supply additional computation services for free during the VC protocol. These extensions are deferred to the full version.

3 Verifiable computation from RE and MAC

3.1 Definitions

Message Authentication Codes. A *one-time message authentication code* (MAC) is an efficiently computable function $\mathtt{MAC} : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ which maps a message $x \in \mathcal{M}$ and a random secret key $k \in \mathcal{K}$ to a signature $\sigma = \mathtt{MAC}_k(x) \in \mathcal{C}$. A MAC is statistically secure with error ε if for every $x \in \mathcal{M}$ and every computationally unbounded adversary A, $\Pr_k[A(x, \mathtt{MAC}_k(x)) = (y, \mathtt{MAC}_k(y)) \wedge (y \neq x)] \leq \varepsilon$.

Verifiable Computation. A *verifiable computation protocol* (VC) for a function f with soundness error $0 \leq \varepsilon \leq 1$ is an interactive protocol between a client C and a server P such that (1) perfect completeness[10]: for every input x the client outputs $f(x)$ at the interaction $(C, P)(x)$ with probability 1; and (2) soundness: for every input x of the client and every cheating P^*, we have $\Pr[(C, P^*)(x) \notin \{f(x), \perp\}] \leq \varepsilon$, where $\perp$ is a special "rejection" symbol and the probability is taken over the coin tosses of the client C. If P^* is restricted to be a polynomial-size circuit then the protocol is *computationally sound*.

Randomized Encoding [23,3]. A *randomized encoding* (RE) for a function f is a non-interactive protocol in which the client uses its randomness r and its input x to compute a message $\hat{y} = \hat{f}(x; r)$ and sends it to the server, who responds by applying a decoder algorithm B to $\hat{y}$, recovers $f(x)$ and sends it back to the client. The protocol should satisfy: (1) *perfect completeness* (as in VC); and (2) ε*-privacy*: There exists a simulator S^* such that for every x the distribution $S^*(1^{|x|}, f(x))$ is at most ε-far (in statistical distance) from the distribution of the client's message $\hat{f}(x; r)$. *Computational privacy* is defined by restricting S^* to be a polynomial-size circuit and replacing statistical distance with ε-computational indistinguishability.

3.2 Our Reduction

Our protocol is described in Figure 1.

The following lemma (whose proof is deferred to the full version) holds both in the statistical and computational setting:

[10] Due to space limitations, we always assume that protocols have perfect completeness. Our results hold in the more general setting where protocols have some completeness error δ.

- Primitives: MAC $\mathtt{MAC}$, and RE $\hat{g}$ for $g(k,x) = \mathtt{MAC}_k(f(x))$.
- Client's input: $x \in \{0,1\}^n$.

1. **Client**: Client chooses a random key k for $\mathtt{MAC}$, and random coins r and sends x together with the encoding $\hat{g}((k,x);r)$.
2. **Server**: Applies the decoder of $\hat{g}((k,x);r)$ and sends the result z together with $y = f(x)$.
3. **Client**: Accepts y if $\mathtt{MAC}_k(y)$ equals to z.

Fig. 1. A verifiable computation protocol for f based on a RE for $g(k,x) = \mathtt{MAC}_k(f(x))$

Lemma 1. *Suppose that the RE and MAC have privacy errors of ε and ε', respectively. Then, the above protocol is a VC for f with soundness error $\varepsilon + \varepsilon'$.*

Proof (sketch). Fix a cheating server P^* and an input x^*. Let α be the probability that the client accepts some $y \neq f(x^*)$. We show that $\alpha \leq \varepsilon + \varepsilon'$. Consider the following attack on the MAC. Given x^* we compute $f(x^*)$ and ask for a signature $\mathtt{MAC}_k(f(x^*))$, where k is an unknown uniformly chosen MAC key. Then, we will use the RE simulator to simulate the encoding of $\mathtt{MAC}_k(f(x))$ up to distance ε and send the result together with x to P^*. Finally, output the pair (y,z) generated by P^*. Since the view of the adversary is ε-close to the view of P^* in a real interaction, the attack succeeds with probability at least $\alpha - \varepsilon$, which by the security of the MAC should be at most ε'. It follows that $\alpha \leq \varepsilon + \varepsilon'$.

Acknowledgements. We thank Guy Rothblum for useful discussions, and Yael Tauman Kalai for sharing with us a copy of [11].

References

1. Abadi, M., Feigenbaum, J., Kilian, J.: On hiding information from an oracle. In: STOC (1987)
2. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Computional Complexity 15(2), 115–162 (2006)
3. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. In: SICOMP, vol. 36(4), pp. 845–888 (2006)
4. Babai, L.: Trading group theory for randomness. In: STOC (1985)
5. Beaver, D., Feigenbaum, J.: Hiding instances in multioracle queries. In: Choffrut, C., Lengauer, T. (eds.) STACS 1990. LNCS, vol. 415. Springer, Heidelberg (1990)
6. Beimel, A., Gál, A.: On arithmetic branching programs. JCSS 59(2), 195–220 (1999)
7. Blum, M., Kannan, S.: Programs that check their work. In: STOC (1989)
8. Blum, M., Luby, M., Rubinfeld, R.: Program result checking against adaptive programs and in cryptographic settings. In: Distributed Computing and Crypthography: DIMACS Workshop (1990)
9. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting programs with applications to numerical problems. In: STOC (1990)

10. Buntrock, G., Damm, C., Hertrampf, U., Meinel, C.: Structure and importance of logspace-MOD-classes. In: Jantzen, M., Choffrut, C. (eds.) STACS 1991. LNCS, vol. 480. Springer, Heidelberg (1991)
11. Chung, K.M., Kalai, Y.T., Vadhan, S.: Improved delegation of computation using fully homomorphic encryption (2010) (in submission)
12. Cramer, R., Fehr, S., Ishai, Y., Kushilevitz, E.: Efficient multi-party computation over rings. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003)
13. Damgård, I., Nielsen, J.: Universally composable efficient multiparty computation from threshold homomorphic encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer, Heidelberg (2003)
14. Feigenbaum, J.: Locally random reductions in interactive complexity theory. In: Advances in Computational Complexity Theory. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 13, pp. 73–98 (1993)
15. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. Cryptology ePrint Archive, Report 2009/547 (2009), http://eprint.iacr.org/
16. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC (2009)
17. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, vol. 7 (1987)
18. Goldwasser, S., Gutfreund, D., Healy, A., Kaufman, T., Rothblum, G.N.: Verifying and decoding in constant depth. In: STOC (2007)
19. Goldwasser, S., Gutfreund, D., Healy, A., Kaufman, T., Rothblum, G.N.: A (de)constructive approach to program checking. In: STOC (2008)
20. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: STOC (2008)
21. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. In: SICOMP, vol. 18 (1989)
22. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 264–282. Springer, Heidelberg (2005)
23. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: FOCS (2000)
24. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: ICALP (2002)
25. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 294–314. Springer, Heidelberg (2009)
26. Kalai, Y.T., Raz, R.: Probabilistically checkable arguments. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009. LNCS, vol. 5677, pp. 143–159. Springer, Heidelberg (2009)
27. Karchmer, M., Wigderson, A.: On span programs. In: Structure in Complexity Theory Conference (1993)
28. Lipton, R.J.: New directions in testing. In: Distributed Computing and Cryptography. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 2, pp. 191–202 (1991)
29. Mahajan, M., Vinay, V.: Determinant: combinatorics, algorithms and complexity. Chicago J. Theoret. Comput. Sci. (5) (1997)
30. Micali, S.: CS proofs (extended abstracts). In: FOCS (1994)
31. Rubinfeld, R.: Designing checkers for programs that run in parallel. Algorithmica 15(4), 287–301 (1996)
32. Yao, A.C.: How to generate and exchange secrets. In: FOCS (1986)

Mergeable Dictionaries

John Iacono* and Özgür Özkan**

Department of Computer Science and Engineering
Polytechnic Institute of NYU
Six MetroTech Center
Brooklyn, NY 11201-3840 USA

Abstract. A data structure is presented for the Mergeable Dictionary abstract data type, which supports the operations Predecessor-Search, Split, and Merge on a collection of disjoint sets of totally ordered data. While in a typical mergeable dictionary (e.g. 2-4 Trees), the Merge operation can only be performed on sets that span disjoint intervals in keyspace, the structure here has no such limitation. A data structure which can handle arbitrary Merge operations in $\mathcal{O}(\log n)$ amortized time in the absence of Split operations was presented by Brown and Tarjan [2]. A data structure which can handle both Split and Merge operations in $\mathcal{O}(\log^2 n)$ amortized time was presented by Farach and Thorup [4]. In contrast, our data structure supports all operations, including Split and Merge, in $O(\log n)$ amortized time, thus showing that interleaved Merge operations can be supported at no additional cost vis-à-vis disjoint Merge operations.

Keywords: data structures, amortized analysis.

1 Introduction

Consider the following operations on a data structure which maintains a dynamic collection $\mathcal{S}$ of disjoint sets $\{S_1, S_2, \ldots\}$ which partition some totally ordered universal set $\mathcal{U}$:

- $p \leftarrow \textsc{Search}(S, x)$: Returns the largest element in S that is at most x.
- $(A, B) \leftarrow \textsc{Split}(S, x)$: Splits S into two sets $A = \{y \in S \,|\, y \leq x\}$ and $B = \{y \in S \,|\, y > x\}$. S is removed from $\mathcal{S}$ while A and B are inserted.
- $C \leftarrow \textsc{Merge}(A, B)$: Creates $C = A \cup B$. C is inserted into $\mathcal{S}$ while A and B are removed.

We call a data structure that supports these operations a *Mergeable Dictionary*. In this paper we present a data structure, which implements these operations in amortized time $\mathcal{O}(\log n)$, where n is the total number of items in $\mathcal{U}$. What makes the concept of a Mergeable Dictionary interesting is that the Merge

* Research supported by NSF grant CCR-0430849 and by a Alfred P. Sloan Fellowship.
** Research supported by US Department of Education grant P200A090157.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 164–175, 2010.

operation does not require that the two sets being merged occupy disjoint intervals in keyspace. A data structure for merging arbitrarily interleaved sets has been presented by Brown and Tarjan [2]. While their structure has an amortized time complexity of $\mathcal{O}(\log n)$ when the SPLIT operation is excluded, in the presence of the SPLIT operation, the amortized time complexity of the structure becomes $\Omega(n)$. A data structure also supporting the SPLIT operation has appeared independently in the context of Union-Split-Find, Mergeable Trees, and string matching in Lempel-Ziv compressed text. In all three cases, a $o(\log^2 n)$ bound on mergeable dictionary operations could not be achieved. We present a data structure that is able to break through this bound with the use of a novel weighting scheme applied to an extended version of the Biased Skip List data structure [1]. We first present a high-level description of the core data structure of the previous work.

1.1 High-Level Description

The basic idea of the structure is simple. Store each set using an existing dictionary that supports SEARCH, SPLIT, and JOIN[1] in $\mathcal{O}(\log n)$ time (e.g. 2-4 trees). Thus, the only operation that requires a non-wrapper implementation is MERGE. One first idea would be to implement MERGE in linear time as in *Merge-Sort*, but this performs poorly, as one would expect. A more intelligent idea is to use a sequence of searches to determine how to partition the two sets into sets of *segments* that span maximal disjoint intervals. Then, use a sequence of SPLITs to split each set into the segments and a sequence JOIN operations to piece together the segments in sorted order. As the number of segments between two sets being merged could be $\Theta(n)$, the worst-case runtime of such an implementation is $\mathcal{O}(n \log n)$, even worse than the $\mathcal{O}(n)$ of a brute-force merge. However, it is impossible to perform many MERGEs with a high number of segments, and an amortized analysis bears this out; there are only $\mathcal{O}(\log n)$ amortized segments per MERGE. Thus, since each segment can be processed in $\mathcal{O}(\log n)$ time, the total amortized cost per MERGE operation is $\mathcal{O}(\log^2 n)$.

In [8], it was shown that there are sequences of operations that have $\Theta(\log n)$ amortized segments per MERGE. This, combined with the worst-case lower bound of $\Omega(\log n)$ for the operations needed to process each segment seemingly gives a strong argument for a $\Omega(\log^2 n)$ lower bound, which was formally conjectured by Lai [8]. It would appear that any effort to circumvent this impediment would require abandoning storing each set in sorted order. We show this is not necessary, as a weighting scheme allows us finesse the balance between the cost of processing each segment, and the number of segments to be processed; we, in essence, prevent the worst-case of these two needed events from happening simultaneously. Our scheme, combined with an extended version of Biased Skip Lists, allows us to speed up the processing of each segment to $\mathcal{O}(1)$ when there are many of them, yet gracefully degrades to the information-theoretically mandated $\Theta(\log n)$ worst-case time when there are only a constant number of segments.

[1] JOIN merges two sets but requires that the sets span disjoint intervals in keyspace.

The details, however, are numerous. In Section 3 we discuss Biased Skip Lists how we extend them. Given this, in Section 4 a full description of our structure is presented, and in Section 5 the runtime is analyzed. Due to space constraints, some of the technical details are deferred to the full version of the paper [7].

1.2 Relationship to Existing Work

The underlying problem addressed here has come up independently three times in the past, in the context of Union-Split-Find [8], Mergeable Trees [6,5], and string matching in Lempel-Ziv compressed text [4]. All three of these results, which were initially done independently of each other, bump up against the same $\mathcal{O}(\log^2 n)$ issue with merging, and all have some variant of the $\mathcal{O}(\log^2 n)$ structure outlined above at their core. While the intricacy of the latter two precludes us claiming here to reduce the squared logarithmic terms in their runtimes, we believe that we have overcome the fundamental obstacle towards this improvement.

2 The Heuristic Yielding the $\mathcal{O}(\log^2 n)$ Amortized Bound

We will describe a heuristic for the MERGE operation presented in [3] and used in previous work [6,5,8], and show that the use of this heuristic yields $\mathcal{O}(\log^2 n)$ amortized bounds as a warm up.

2.1 The Segment Merging Heuristic

Define a *segment* of the MERGE(A, B) operation to be a maximal subset S of either set A or set B such that no element in $(A \cup B) \setminus S$ lies in the interval $[\min(S), \max(S)]$. Each set in the collection is stored as a balanced search tree (i.e. 2-4 tree) with level links. The FIND, SEARCH, and SPLIT operations are implemented[2] in a standard way to run in $\mathcal{O}(\log n)$ worst-case time. The MERGE(A, B) operation is performed as follows: first locate the minimum and maximum element of each segment of the MERGE(A, B) operation using the SEARCH operation and the level links, then extract all these segments using the SPLIT operation, and finally we merge all the segments in the obvious way using the standard JOIN operation. Therefore since each operation takes $\mathcal{O}(\log n)$ worst-case time, the total running time is $\mathcal{O}(T \cdot \log n)$ where T is the number of segments. We now analyze all the operations using the potential method [10], with respect to two parameters: $n = |\mathcal{U}|$, and m, the total number of all operations. Let D_i represent the data structure after operation i, where D_0 is the initial data structure. Operation i has a cost of c_i and transforms D_{i-1} into D_i. We will define a potential function $\Phi : \{D_i\} \to \mathbb{R}$ such that $\Phi(D_0) = 0$ and $\Phi(D_i) \geq 0$ for all i. The amortized cost of operation i, $\hat{c}_i$, with respect to Φ is defined as $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$. The total amortized cost of m operations will

[2] See [8] for a detailed description of this implementation.

be $\sum_{i=1}^{m} \hat{c_i} = \sum_{i=1}^{m} (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \sum_{i=1}^{m} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{m} c_i$ since $\Phi(D_n) \geq 0$ and $\Phi(D_0) = 0$. Thus, the amortized cost will give us an upper bound on the worst-case cost.

Next, we describe a potential function which yields an amortized bound of $\mathcal{O}(\log^2 n)$ on the running time. This potential function was essentially used in [6,5,4] which are the only instances where a $o(n)$ solution has been presented.

2.2 The Potential Function

We need to define some terminology before describing the potential function. Let $\text{pos}_S(x)$ be the position of x in set S, or more formally $pos_S(x) = |\{y \in S \mid y \leq x\}|$. Then $g_S(k)$, the size of the k^{th} gap of set S, is the difference of positions between the element of position k and $k+1$ of set S in universe U. In other words, $g_S(k) = pos_U(x) - pos_U(y)$ where $pos_S(x) = k$ and $pos_S(y) = k+1$. For the boundary cases, let $g_S(0) = g_S(|S|) = 1$. Recall that D_i is the data structure containing our dynamic collection of disjoint sets, $\mathcal{S}^{(i)} = \{S_1^{(i)}, S_2^{(i)}, \ldots\}$ after the i^{th} operation. Then let $\varphi(S) = \sum_{j=1}^{|S|-1} \log g_S(j)$. Finally, we define the potential after the i^{th} operation as follows: $\Phi(D_i) = \kappa_a \cdot \sum_{S \in \mathcal{S}^{(i)}} \varphi(S) \log n$ where κ_a is a positive constant to be determined later. Note that since the collection of sets initially consists of the n singleton sets, the data structure initially has 0 potential ($\Phi(D_0) = 0$). Furthermore, because any gap has size at least 1, the data structure always has non-negative potential ($\Phi(D_i) \geq 0,\ \forall i \geq 0$).

2.3 The Amortized $\mathcal{O}(\log^2 n)$ Bound

The SEARCH, and SPLIT operations have worst-case $\mathcal{O}(\log n)$ running times. The SEARCH operation does not change the structure and therefore does not affect the potential. Observe that the SPLIT operation can only decrease the potential. Thus, the amortized cost of these operations is $\mathcal{O}(\log n)$. Now, suppose the i^{th} operation is MERGE(A, B) where A and B are sets in D_{i-1}. Assume w.l.o.g. that the minimum element in $A \cup B$ is an element of A. Let $I(A, B) = \{A_1, B_1, A_2, B_2, \ldots\}$ be the set of segments of operation MERGE(A, B), where $\max(A_i) < \min(A_j)$ and $\max(B_i) < \min(B_j)$ for $i < j$, and $\max(A_i) < \min(B_i) < \max(B_i) < \min(A_{i+1})$ for all i. As previously noted, the worst-case cost of the MERGE operation is $\mathcal{O}(|I(A, B)| \cdot \log n)$. Let a_i be the size of the gap between the maximum element of A_i and the minimum element of A_{i+1}, or more formally let $a_i = g_{A_i \cup A_{i+1}}(|A_i|)$. Define b_i similarly. Now, let $a'_i = g_{A_i \cup B_i}(|A_i|)$ $a''_i = g_{B_i \cup A_{i+1}}(|B_i|)$ and $b'_i = g_{B_i \cup A_{i+1}}(|B_i|)$ $b''_i = g_{A_{i+1} \cup B_{i+1}}(|A_{i+1}|)$ Note that $a''_i = b'_i$ and $a'_i = b''_{i-1}$. During the analysis we will take into account whether $|I(A, B)|$ is odd or even. Let $\sigma = |I(A, B)| \bmod 2$ and $R = \lfloor (|I(A, B)| - 2)/2 \rfloor$. We have $\hat{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1})$ where $\Phi(D_{i-1}) = \sum_{S \in \mathcal{S} \setminus \{A,B\}} \varphi(S) \kappa_a \log n + (\sum_i \varphi(A_i) + \sum_i \varphi(B_i)) \kappa_a \log n + \left(\sum_{i=1}^{R} (\log a_i + \log b_i) + \sigma \log a_{R+1} \right) \kappa_a \log n$, and $\Phi(D_i) = \sum_{S \in \mathcal{S} \setminus \{A,B\}} \varphi(S) \kappa_a \log n + (\sum_i \varphi(A_i) + \sum_i \varphi(B_i) + \log a'_1) \kappa_a \log n + \frac{1}{2} \left(\sum_{i=1}^{R} \left(\log a''_i + \log b'_i + \log b''_i + \log a'_{i+1} \right) + \sigma \left(\log a''_{R+1} + \log b'_{R+1} \right) \right) \kappa_a \log n$.

This gives us $\Phi(D_i) - \Phi(D_{i-1}) \leq \frac{1}{2}\left(\sum_{i=1}^{R} \log a_i'' b_i' b_i'' a_i' - 2\log a_i b_i\right) k \log n + \mathcal{O}(\log^2 n) \leq -k' \cdot I(A,B) k \log n + \mathcal{O}(\log^2 n)$. This combined with the actual cost of the MERGE operation yields the $\mathcal{O}(\log^2 n)$ amortized cost for MERGE. Thus, the amortized cost of the MERGE operation is $\mathcal{O}(\log^2 n)$. Combined with the arguments before, this gives us the following theorem:

Theorem 1. *The Mergeable Dictionary problem can be solved such that a sequence of m* SEARCH, SPLIT, *and* MERGE *operations can be executed in $\mathcal{O}(m \log^2 n)$ worst-case time.*

3 Biased Skip Lists with Extended Operations

Biased skip lists [1] are a variant of skip lists [9] which we assume the reader is familiar with. In order to be able to design a highly tuned MERGE operation, we will extend biased skip lists to support finger split, finger join, and finger reweight operations. First, we describe the essential details of biased skip lists[3].

3.1 Biased Skip Lists

We will first cover basic definitions followed by the three key invariants of the structure.

Definitions. A biased skip list (BSL) S stores an ordered set X where each element $x \in X$ corresponds to a node[4] $x \in S$ with *weight* $w(x)$, which is user-defined, and integral *height* $h(x)$, which is initially computed from the weight of the node. For our purposes, we will assume that the weights are bounded from below by 1 and bounded from above by a polynomial in n. Each node $x \in S$ is represented by an array of length $h(x)+1$ called the *tower* of node x. The *level-j predecessor*, $L_j(x)$, of x is the largest node k in S such that $k < x$ and $h(k) \geq j$. The *level-j successor*, $R_j(x)$, is defined symmetrically. The j^{th} element of the tower of node x, contains pointers to the j^{th} elements of towers of node $L_j(x)$ and node $R_j(x)$ with the exception of towers of adjacent nodes where pointers between any pair of adjacent nodes x and y on level $\min(h(x), h(y)) - 1$ are nil and the pointers below this level are undefined. Node levels progress from top to bottom. Two distinct elements x and y are called *consecutive* if and only if they linked together in S; or equivalently if and only if for all $x < z < y$, $h(z) < \min(h(x), h(y))$. A *plateau* is a maximal set of consecutive nodes of the same height. The *rank* of a node x is defined as $r(x) = \lfloor \log_a w(x) \rfloor$ where a is a constant. For our purposes, we will set $a = 2$. Additionally, let $\text{pred}_X(x)$ be the predecessor of x in set X, and let $\text{succ}_X(x)$ be the successor of x in set X. Let $H(X) = \max_{x \in X} h(x)$. Let $S[\hookleftarrow j] = \{x \in S \mid x \leq j\}$ and $S[j\rightarrow] = \{x \in S \mid x > j\}$. Let $W(S) = \sum_{x \in S} w(x)$. Also let $W_{[i,j]}(S) = \sum_{x \in S; i \leq x \leq j} w(x)$. For convenience, we imagine sentinel nodes $-\infty$ and $+\infty$ of height $H(S)$ at the beginning and end of biased skip list S. These sentinels are not actually stored or maintained.

[3] The reader is referred to [1] for further details on biased skip lists.

[4] We will use the terms "element", "node", "key", and "item" interchangeably; the context clarifies any ambiguity.

Invariants. The three invariants of biased skip lists are listed below. Note that a and b can be suitable constants satisfying the definition of (a, b)-biased skip lists. For our purposes it is sufficient to set $a = 2, b = 6$.

Definition 1. *For any a and b such that $1 < a \leq \lfloor \frac{b}{3} \rfloor$, an (a, b)-biased skip list is a biased skip list with the following properties:*

(**I0**) *Each item x has height $h(x) \geq r(x)$.*
(**I1**) *There are never more than b consecutive items of any height.*
(**I2**) *For each node x and for all i such that $r(x) < i \leq h(x)$, there are at least a nodes of height $i-1$ between x and any consecutive node of height at least i.*

In the remainder of the paper, we will refer to $(2, 6)$-biased skip lists simply as biased skip lists.

3.2 Operations

We now describe the original biased skip list operations we will be using in our data structure.

- $p \leftarrow \textsc{BslSrc}(S, i)$: Performs a standard search in biased skip list S using search key i. This operation runs in worst-case $\mathcal{O}(\log n)$ time.
- $p \leftarrow \textsc{BslFSrc}(i,j)$: Starting from a given finger to a node i in some biased skip list S performs a predecessor search in S using j as the search key in $\mathcal{O}\left(1 + \log \frac{W_{[i,\mathrm{succ}_S(j)]}(S)}{\min(w(i), w(\mathrm{pred}_S(j)), w(\mathrm{succ}_S(j)))}\right)$ worst-case time.
- $(A, B) \leftarrow \textsc{BslSplit}(S, i)$: Splits the biased skip list S at i into two biased skip lists A and B storing sets $\{x \in S \mid x \leq i\}$ and $\{x \in S \mid x > i\}$ respectively. This operation runs in worst-case $\mathcal{O}(\log n)$ time.
- $\textsc{BslRew}(S, i, w)$: Changes the weight of node $i \in S$ to w. This operation runs in worst-case $\mathcal{O}(\log n)$ time.

3.3 Extended Operations

We now describe the extended biased skip list operations we will be using in our data structure. Due to limited space, we defer the full treatment of the extended operations, which have straightforward implementations, to the full version of the paper [7], and only list the operations here along with their running times.

- $(A, B) \leftarrow \textsc{BslFSplit}(f)$: Given a pointer to node $f \in S$, $\textsc{BslFSplit}(f)$ splits biased skip list S into two biased skip lists, A and B, storing sets $\{x \in S \mid x \leq f\}$ and $\{x \in S \mid x > f\}$ respectively, and returns an ordered pair of handles to A and B.
- $S \leftarrow \textsc{BslFJoin}(\ell, r)$: Given pointers to ℓ and r, the maximum and minimum nodes of two distinct biased skip lists A and B respectively, $\textsc{BslFJoin}(\ell, r)$ returns a new biased skip list C containing all elements of A and B assuming $\ell < r$. A and B are destroyed in the process.
- $\textsc{BslFRew}(f, w)$: Given a pointer to a node $f \in S$, changes its weight to w while preserving invariants (**I0**), (**I1**), and (**I2**) of the biased skip list containing f.

3.4 The Analysis of Extended Operations

We now give upper bounds on the worst-case running times of the extended operations we described above. The proofs are omitted.

Lemma 1. *The* $(A, B) \leftarrow$ BslFSplit(f) *operation, where* $f \in S$*, has an amortized time complexity of* $\mathcal{O}(\min(H(A'), H(B')) - \min(r(\max(A')), r(\min(B'))) + 1)$ *where* $A' = S[\leftarrow f]$ *and* $B' = S[f \rightarrow]$.

Lemma 2. *Given a set* $L_0 = \{\Lambda_1, \Lambda_2, \ldots, \Lambda_m\}$ *of biased skip lists, a sequence of* $(S_k, T_k) \leftarrow$ BslFSplit(f_k) *operations for* $1 \leq k \leq t$ *where* $f_k \in U_k$*,* $U_k \in L_{k-1}$*,* $S'_k = U_k[\leftarrow f_k]$*, and* $T'_k = U_k[f_k \rightarrow]$ *can be executed in worst-case time* $\mathcal{O}(\sum_{i=1}^{m}(H(\Lambda_i) - \min(h(\min(\Lambda_i)), h(\max(\Lambda_i))) + 1) + \sum_{k=1}^{t}(\min(H(S'_k), H(T'_k)) - \min(r(\max(S'_k)), r(\min(T'_k))) + 1))$.

Lemma 3. *The* $S \leftarrow$ BslFJoin(ℓ, r) *operation, where* $\ell \in A, r \in B$*, has an amortized time complexity of* $\mathcal{O}(\min(H(A), H(B)) - \min(h(\max(A)), h(\min(B))) + 1)$.

Lemma 4. *Given a set* $L_0 = \{\Lambda_1, \Lambda_2, \ldots, \Lambda_m\}$ *of biased skip lists, a sequence of* $U_k \leftarrow$ BslFJoin(ℓ_k, r_k) *operations for* $1 \leq k \leq t$ *where* $\ell_k \in S_k$*,* $r_k \in T_k$ *and* $S_k, T_k \in L_{k-1}$ *can be executed in worst-case time* $\mathcal{O}(\sum_{i=1}^{m}(H(\Lambda_i) - \min(h(\min(\Lambda_i)), h(\max(\Lambda_i))) + 1) + \sum_{k=1}^{t}(\min(H(S_k), H(T_k)) - \min(h(\max(S_k)), h(\min(T_k))) + 1))$.

Lemma 5. *The* BslFRew(f, w) *operation, where* $f \in S$*, has a worst-case and amortized time complexity of* $\mathcal{O}(\max(H(S), r'(f)) - \min(h(f), r'(f)) + 1)$.

Lemma 6. *The* BslFSrc*,* BslFSplit*,* BslFJoin*, and* BslFRew *operations all have a worst-case time complexity of* $\mathcal{O}(\log n)$.

4 Our Data Structure: The Mergeable Dictionary

The Mergeable Dictionary stores each set in the collection $\mathcal{S}$ as a biased skip list. The weight of each node in each biased skip list is determined by $\mathcal{S}$. When the collection of sets is modified, for instance via a Merge operation, in order to reflect this change in the data structure, besides splitting and joining biased skip lists, we need to ensure the weights of the affected nodes are properly updated and biased skip list invariants (**I0**), (**I1**), and (**I2**) are preserved. For simplicity we assume that D_0 is the collection of singleton sets. This lets us precompute, for each node x, $\text{pos}_{\mathcal{U}}(x)$, the global position of x. For the Merge algorithm, we will use the same basic approach outlined in Section 2, the segment merging heuristic, which works by extracting the segments from each set and then gluing them together to form the union of the two sets. Before we discuss the implementation of each operation in detail, we need to describe the weighting scheme.

4.1 Weighting Scheme

Let the weight of a node x, $w(x)$, be the sum of the sizes of its adjacent gaps. In other words, if $pos_S(x) = k$ for some node $x \in S$, then we have $w(x) = g_S(k-1) + g_S(k)$. Recall that $g_S(0) = g_S(|S|) = 1$. Observe that this implies for any set S, $W(S) \leq 2n$.

4.2 The Search and Split Operations

The SEARCH(X, i) operation can be performed by simply invoking BSLSRC(X, i). The SPLIT(X, i) operation can be performed by simply invoking BSLSPLIT(X, i) and running BSLFREW on one node in each of the resulting biased skip lists to restore the weights.

4.3 The Merge Operation

The MERGE(A, B) operation can be viewed as having four essential phases: finding the segments, extracting the segments, updating the weights, and gluing the segments. A more detailed description follows.

Phase I: Finding the segments. Assume $\min(A) < \min(B)$ w.l.o.g. Let $z = \lceil |I(A,B)|/2 \rceil$ and $v = \lfloor |I(A,B)|/2 \rfloor$. Recall that $I(A,B) = \{A_1, B_1, A_2, B_2, \ldots\}$ is the set of segments associated with the MERGE(A, B) operation where A_i and B_i are the i^{th} segment of A and B respectively. We have $\min(A_1) = \min(A)$ and $\min(B_1) = \min(B)$. Given $\min(A_i)$ and $\min(B_i)$, we find $\max(A_i)$ by invoking BSLFSRC($\min(A_i)$,$\min(B_i)$). Similarly, given $\min(B_i)$ and $\min(A_{i+1})$ we find $\max(B_i)$ by invoking BSLFSRC($\min(B_i)$,$\min(A_{i+1})$). Lastly, given $\max(A_i)$ and $\max(B_i)$, observe that $\min(A_{i+1}) = \text{succ}_A(\max(A_i))$ and $\min(B_{i+1}) = \text{succ}_B(\max(B_i))$. Note that the succ() operation is performed in constant time in a biased skip list using the lowest successor link of a node. At the end of this phase, all the segments are found. Specifically, we have computed for all i and j ($\min(A_i)$, $\max(A_i)$) and ($\min(B_j)$, $\max(B_j)$).

Phase II: Extracting the segments. Since we know where the minimum and maximum node of each segment is from the previous phase, we can extract all the segments easily in order by invoking BSLFSPLIT($\max(A_i)$) for $1 \leq i < z$, and BSLFSPLIT($\max(B_j)$) for $1 \leq j < v$.

Phase III: Updating Weights. Next, we need to update the weights of the affected nodes. Let the new weight of item x be $w'(x)$. Then for $2 \leq i \leq z$, let $w'(\min(A_i)) = w(\min(A_i)) + pos_U(\max(A_{i-1})) - pos_U(\max(B_{i-1}))$, for $2 \leq i \leq v$, let $w'(\min(B_i)) = w(\min(B_i)) + pos_U(\max(B_{i-1})) - pos_U(\max(A_i))$, for $1 \leq i < z$, let $w'(\max(A_i)) = w(\max(A_i)) + pos_U(\min(B_i)) - pos_U(\min(A_{i+1}))$, for $1 \leq i < v$, let $w'(\max(B_i)) = w(\max(B_i)) + pos_U(\min(A_{i+1})) - pos_U(\min(B_{i+1}))$. We also have $w'(\min(B_1)) = w(\min(B_1)) - 1 + pos_U(\min(B_1)) - pos_U(\max(A_1))$. If $|I(A,B)|$ is even, we have $w'(\max(A_z)) = w(\max(A_z)) - 1 + pos_U(\min(B_z)) -$

$pos_U(\max(A_z))$. If $|I(A,B)|$ is odd, we have $w'(\max(B_v)) = w(\max(B_v)) - 1 + pos_U(\min(A_z)) - pos_U(\max(B_v))$.

We can perform these updates by invoking BslFRew($\min(A_i)$,$w'(\min(A_i))$) for $2 \leq i \leq z$, invoking BslFRew($\max(A_i)$,$w'(\max(A_i))$) for $1 \leq i \leq v$, invoking BslFRew($\min(B_j)$,$w'(\min(B_j))$) for $1 \leq j \leq v$, and lastly invoking BslFRew($\max(B_j)$,$w'(\max(B_j))$) for $1 \leq j < z$.

Phase IV: Gluing the segments. Since we assumed w.l.o.g. that $\min(A) < \min(B)$*, the correct order of the segments is* $(A_1, B_1, A_2, B_2, \ldots)$ *by construction. We can glue all the segments by invoking* BslFJoin($\max(A_i)$,$\min(B_i)$) *for* $1 \leq i \leq v$ *and* BslFJoin($\max(B_i)$,$\min(A_{i+1})$) *for* $1 \leq i < z$.

5 Analysis of the Mergeable Dictionary

Before we can analyze the amortized time complexity of the Mergeable Dictionary operations, we need a new potential function.

5.1 The New Potential Function

Let D_i be the data structure containing our dynamic collection of disjoint sets, $\mathcal{S}^{(i)} = \{S_1^{(i)}, S_2^{(i)}, \ldots\}$ after the i^{th} operation. Let $\varphi(S) = \sum_{x \in S}(\log g_S(pos_S(x) - 1) + \log g_S(pos_S(x)))$. Then we define the potential after the i^{th} operation as $\Phi(D_i) = \kappa_d \cdot \sum_j \varphi(S_j^{(i)})$ where κ_d is a constant to be determined later. Note that the main difference between this function and the one in Section 2.2 is the elimination of the $\log n$ term.

5.2 The Analysis of the Search and Split Operations

We now show that all the operations except Merge have a worst-case time complexity of $\mathcal{O}(\log n)$, and they do not cause a substantial increase in the potential which yields that their amortized time complexity is also $\mathcal{O}(\log n)$.

Theorem 2. *The worst-case and amortized time complexity of the* Search(S, x) *operation and the* Split(S, x) *operation is* $\mathcal{O}(\log n)$.

Proof. The worst-case time complexity of the operations BslFSrc, BslSplit, and BslRew invoked by the Search and Split operations is $\mathcal{O}(\log n)$ by Lemma 6. Recall that since Search does not change the structure, the potential remains the same; and Split can only decrease the potential. Therefore, worst-case and amortized time complexity of Search and Split is $\mathcal{O}(\log n)$.

5.3 The Analysis of the Merge Operation

All we have left to do is show that the amortized time complexity of the Merge operation is $\mathcal{O}(\log n)$. We define $F(A_i)$ and $F(B_j)$ next.

Definition 2. *Consider the* MERGE(A, B) *operation. Recall that* $w'(x)$ *is the new weight of node* x *after the* MERGE(A, B) *operation. Then, for* $1 < i < z$, *let* $F(A_i) = \log \frac{w(\max(A_{i-1}))+w(\min(A_{i+1}))+\sum_{x\in A_i} w(x)}{\min(w'(\max(B_{i-1})),w'(\min(A_i)),w'(\max(A_i)),w'(\min(B_i)))}$ *and for* $1 < j < v$, *let* $F(B_j) = \log \frac{w(\max(B_{j-1}))+w(\min(B_{j+1}))+\sum_{x\in B_j} w(x)}{\min(w'(\max(A_j)),w'(\min(B_j)),w'(\max(B_j)),w'(\min(A_{j+1})))}$. *For the boundary cases, let* $F(A_1) = F(B_1) = F(A_z) = F(B_v) = \log n$.

The Worst-Case Time Complexity. We need to bound the worst-case time complexity of each phase of the MERGE(A, B) operation. We will need the following lemma to bound the worst-case time complexity of Phases II-IV.

Lemma 7. *Given a biased skip list* S *and any node* $f \in S$, *recall that* $S[\hookleftarrow f] = \{x \in S \mid x \le f\}$. *Then,* $H(S[\hookleftarrow f]) \le \log W(S[\hookleftarrow f])$.

Proof. Let $R = \max_{x\in S[\hookleftarrow f]} r(x)$ and $N_r(t) = \{x \in S[\hookleftarrow f] \mid r(x) = t\}$. Also, let $N_h(t) = \{x \in S[\hookleftarrow f] \mid h(x) \ge t, r(x) \le t\}$. Then we have $W(S[\hookleftarrow f]) \ge \sum_{i=2}^{R} 2^i N_r(i) + \sum_{\substack{x\in S[\hookleftarrow f],\\ r(x)=1}} w(x) \ge \sum_{i=2}^{R} 2^i N_r(i) + 2N_h(1)$. Due to (**I2**), we have

$$\begin{aligned} N_h(i) &\ge 2(N_h(i+1) - N_r(i+1)) \ge 2^{R-1} N_h(R) - \sum_{i=2}^{R} 2^{i-1} N_r(i) \\ &\ge 2^{H(S[\hookleftarrow f])-1} N_h(H(S[\hookleftarrow f])) - \sum_{i=2}^{R} 2^{i-1} N_r(i) \quad (N_r(t) = 0 \text{ for } t > R) \\ &\ge 2^{H(S[\hookleftarrow f])-1} - \sum_{i=2}^{R} 2^{i-1} N_r(i) \end{aligned}$$

which yields

$$\begin{aligned} W(S[\hookleftarrow f]) &\ge \sum_{i=2}^{R} 2^i N_r(i) + 2N_h(1) \\ &\ge \sum_{i=2}^{R} 2^i N_r(i) + 2\left(2^{H(S[\hookleftarrow f])-1} - \sum_{i=2}^{R} 2^{i-1} N_r(i)\right) \\ &\ge 2^{H(S[\hookleftarrow f])} \\ \log W(S[\hookleftarrow f]) &\ge H(S[\hookleftarrow f]). \end{aligned}$$

Theorem 3. *The* MERGE(A, B) *operation has a worst-case time complexity of* $\mathcal{O}\left(\log n + \sum_{i=2}^{z-1} F(A_i) + \sum_{j=2}^{v-1} F(B_j)\right)$.

Proof. Note that $w'(\min(B_i)) < w(\min(A_{i+1}))$ and $w'(\min(B_i)) < w(\max(A_i))$. The worst-case time complexity of the MERGE(A, B) operation is determined by the time it spends on each of the four phases. Therefore, the theorem follows by the definitions of $F(A_i)$ and $F(B_j)$, and Lemmas 2, 4, 5, 6, and 7.

Amortized Time Complexity. Before we can show that the amortized time complexity of the MERGE(A, B) operation is $\mathcal{O}(\log n)$, we will need to prove three lemmas. Let us first define the potential loss associated with a gap. Recall the definitions of gaps a_i, a'_i, a''_i and similarly b_j, b'_j, b''_j first defined in Section 2.3.

Definition 3. *We define $pl(a_i)$ and $pl(b_j)$, the potential loss associated with gap a_i and the potential loss associated with gap b_i respectively, for $1 \leq i < z$ and $1 \leq j < v$, as follows: $pl(a_i) = 2\log a_i - \log a'_i - \log a''_i$ and $pl(b_j) = 2\log b_j - \log b'_j - \log b''_j$. Assume w.l.o.g. that $\min(A) < \min(B)$. Then let $pl(a_0) = 0$ and $pl(b_0) = -\log a'_1$. If $\max(A) > \max(B)$, then $pl(a_z) = 0$ and $pl(b_v) = -\log a''_{z-1}$. Otherwise, if $\max(A) < \max(B)$, then $pl(b_v) = 0$ and $pl(a_z) = -\log b''_{v-1}$. Note that the potential loss associated with operation MERGE(A, B) is κ_d times the sum of all $pl(a_i)$ and $pl(b_j)$, where κ_d is the constant in the potential function.*

Lemma 8. *Consider gap a_i for any $1 \leq i < z$. Let $a_i^+ = \max(a'_i, a''_i)$ and $a_i^- = \min(a'_i, a''_i)$. Then, $2^{-pl(a_i)} \leq a_i/a_i^+ \leq a_i/a_i^- \leq 2^{pl(a_i)}$ and $2^{-pl(a_i)} \leq a_i^+/a_i^- \leq 2^{pl(a_i)}$. Similarly, for gap b_j for any $1 \leq j < v$, where $b_j^+ = \max(b'_j, b''_j)$ and $b_j^- = \min(b'_j, b''_j)$, we have $2^{-pl(b_j)} \leq b_j/b_j^+ \leq b_j/b_j^- \leq 2^{pl(b_j)}$ and $2^{-pl(b_j)} \leq b_j^+/b_j^- \leq 2^{pl(b_j)}$.*

Proof. Follows directly from Definition 3.

Lemma 9. *Let $I(A, B) = \{A_1, B_1, A_2, B_2, \ldots\}$ be the set of segments with respect to operation MERGE(A, B). For any i and j, where $1 < i < z$ and $1 < j < v$, let $\alpha_i = \max\left(2^{pl(a_{i-2})}, 2^{pl(b_{i-2})}, 2^{pl(a_{i-1})}, 2^{pl(b_{i-1})}, 2^{pl(a_i)}, 2^{pl(b_i)}, 2^{pl(a_{i+1})}\right)$ and $\beta_j = \max\left(2^{pl(b_{j-2})}, 2^{pl(a_{j-1})}, 2^{pl(b_{j-1})}, 2^{pl(a_j)}, 2^{pl(b_j)}, 2^{pl(a_{j+1})}, 2^{pl(b_{j+1})}\right)$. Then, it holds that $F(A_i) = \mathcal{O}(\log \alpha_i)$ and $F(B_j) = \mathcal{O}(\log \beta_j)$.*

Proof. We will present the proof of the first equality. The proof of the second one is analogous. Let $a''_{i-1} = b'_{i-1} = x$. Then, by Lemma 8 we have $\frac{x}{\alpha_i} \leq a'_{i-1}, a'_i, b''_{i-2}, b''_{i-1} \leq x\alpha_i$, $\frac{x}{\alpha_i^2} \leq a''_{i-2}, a''_i, b'_{i-2}, b'_i \leq x\alpha_i^2$, $\frac{x}{\alpha_i^3} \leq a'_{i-2}, a'_{i+1}, b''_i \leq x\alpha_i^3$, and $\frac{x}{\alpha_i^4} \leq a''_{i+1} \leq x\alpha_i^4$ Similarly, by Lemma 8, we have $\frac{x}{\alpha_i} \leq a_{i-1}, b_{i-1} \leq x\alpha_i$, $\frac{x}{\alpha_i^2} \leq a_i, b_{i-2} \leq x\alpha_i^2$, $\frac{x}{\alpha_i^3} \leq a_{i-2}, b_i \leq x\alpha_i^3$, and $\frac{x}{\alpha_i^4} \leq a_{i+1} \leq x\alpha_i^4$. We proceed as follows. For $1 < i < z$, using the inequalities above, we have

$$\begin{aligned} F(A_i) &= \log \frac{w(\max(A_{i-1})) + w(\min(A_{i+1})) + \sum_{x \in A_i} w(x)}{\min(w'(\max(B_{i-1})), w'(\min(A_i)), w'(\max(A_i)), w'(\min(B_i)))} \\ &\leq \log \frac{a_{i-2} + b_{i-2} + a_{i-1} + a_i + b_i + a_{i+1} + 2b_{i-1}}{\min(b'_{i-1}, a''_{i-1}, a'_i, b''_{i-1})} = \mathcal{O}\left(\log \frac{x\alpha_i^4}{x/\alpha_i}\right) \\ &= \mathcal{O}(\log \alpha_i). \end{aligned}$$

The proof of the second equality, $F(B_j) = \mathcal{O}(\log \beta_j)$ for $1 < j < v$, is analogous.

Lemma 10. *For $1 < i < z$ and $1 < j < v$, we have*

$$7 \cdot \sum_i \log 2^{pl(a_i)} + 7 \cdot \sum_j \log 2^{pl(b_j)} > \sum_i \log \alpha_i + \sum_j \log \beta_j.$$

Proof. Observe that a gap a_k can be mapped to at most seven times by unique α_i's and β_j's; namely only by $\alpha_{k-1}, \beta_{k-1}, \alpha_k, \beta_k, \alpha_{k+1}, \beta_{k+1}, \alpha_{k+2}$. Similarly, a gap b_k can be mapped to at most seven times by unique α_i's and β_j's; namely only by $\beta_{k-1}, \alpha_k, \beta_k, \alpha_{k+1}, \beta_{k+1}, \alpha_{k+2}, \beta_{k+2}$. The lemma follows.

Theorem 4. *The* MERGE(A, B) *operation has an amortized time complexity of* $\mathcal{O}(\log n)$.

Proof. We will analyze the MERGE operation using the potential method [10]. Recall that D_i represent the data structure after operation i, where D_0 is the initial data structure. The amortized cost of operation i is $\hat{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1})$. By Theorem 3, we have $c_i = \kappa_e \left(\log n + \sum_{i=2}^{z-1} F(A_i) + \sum_{j=2}^{v-1} F(B_j)\right)$, and by the definitions of $\mathrm{pl}(a_i)$ and $\mathrm{pl}(b_j)$ we have $\Phi(D_i) - \Phi(D_{i-1}) = \kappa_d \cdot \left(\sum_{i=0}^{z} \mathrm{pl}(a_i) + \sum_{j=0}^{v} \mathrm{pl}(b_j)\right)$. Then, applying Lemmas 9 and 10 and setting κ_d appropriately yields that the amortized time complexity of the MERGE(A, B) operation is $\mathcal{O}(\log n)$.

We can now state our main theorem.

Theorem 5. *The Mergeable Dictionary executes a sequence of* m SEARCH, SPLIT, *and* MERGE *operations in worst-case* $\mathcal{O}(m \log n)$ *time.*

Proof. Follows directly from Theorem 2 and Theorem 4.

References

1. Bagchi, A., Buchsbaum, A.L., Goodrich, M.T.: Biased skip lists. Algorithmica 42(1), 31–48 (2005)
2. Brown, M.R., Tarjan, R.E.: A fast merging algorithm. J. ACM 26(2), 211–226 (1979)
3. Demaine, E.D., López-Ortiz, A., Munro, J.I.: Adaptive set intersections, unions, and differences. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 743–752 (2000)
4. Farach, M., Thorup, M.: String matching in lempel-ziv compressed strings. Algorithmica 20(4), 388–404 (1998)
5. Georgiadis, L., Kaplan, H., Shafrir, N., Tarjan, R.E., Werneck, R.F.F.: Data structures for mergeable trees. In: CoRR, abs/0711.1682 (2007)
6. Georgiadis, L., Tarjan, R.E., Werneck, R.F.F.: Design of data structures for mergeable trees. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 394–403. ACM Press, New York (2006)
7. Iacono, J., Özkan, Ö.: Mergeable Dictionaries. In: CoRR, abs/1002.4248 (2010)
8. Lai, K.J.: Complexity of union-split-find problems. Master's thesis, Massachusetts Institute of Technology, Erik Demaine, Adviser (2008)
9. Pugh, W.: Skip lists: A probabilistic alternative to balanced trees. Communications of the ACM 33(6), 668–676 (1990)
10. Tarjan, R.E.: Amortized computational complexity. SIAM Journal on Algebraic Discrete Methods 6(2), 306–318 (1985)

Faster Algorithms for Semi-matching Problems (Extended Abstract)*

Jittat Fakcharoenphol[1], Bundit Laekhanukit[2], and Danupon Nanongkai[3]

[1] Kasetsart University, Bangkok, 10900, Thailand
jittat@gmail.com
http://www.cpe.ku.ac.th/~jtf
[2] University of Waterloo, Waterloo ON, N2L 3G1, Canada
blaekhan@uwaterloo.ca
http://www.math.uwaterloo.ca/~blaekhan
[3] Georgia Institute of Technology, Atlanta GA, 30332
danupon@cc.gatech.edu
http://www.cc.gatech.edu/~danupon

Abstract. We consider the problem of finding *semi-matching* in bipartite graphs which is also extensively studied under various names in the scheduling literature. We give faster algorithms for both weighted and unweighted case.

For the weighted case, we give an $O(nm \log n)$-time algorithm, where n is the number of vertices and m is the number of edges, by exploiting the geometric structure of the problem. This improves the classical $O(n^3)$ algorithms by Horn [Operations Research 1973] and Bruno, Coffman and Sethi [Communications of the ACM 1974].

For the unweighted case, the bound could be improved even further. We give a simple divide-and-conquer algorithm which runs in $O(\sqrt{n}m \log n)$ time, improving two previous $O(nm)$-time algorithms by Abraham [MSc thesis, University of Glasgow 2003] and Harvey, Ladner, Lovász and Tamir [WADS 2003 and Journal of Algorithms 2006]. We also extend this algorithm to solve the *Balance Edge Cover* problem in $O(\sqrt{n}m \log n)$ time, improving the previous $O(nm)$-time algorithm by Harada, Ono, Sadakane and Yamashita [ISAAC 2008].

1 Introduction

In this paper, we consider a relaxation of the maximum bipartite matching problem called *semi-matching* problem, in both weighted and unweighted case. This problem has been previously studied in the scheduling literature under different names, mostly known as (nonpreemptive) scheduling independent jobs on unrelated machines to minimize flow time, or $R||\sum C_j$ in the standard scheduling notation [2,20].

Informally, the problem can be explained by the following off-line load balancing scenario: We are given a set of jobs and machines. Each machine can process

* Full version can be found in [9].

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 176–187, 2010.

one job at a time and it takes different amounts of time to process different jobs. Each job also requires different processing times if processed by different machines. One natural goal is to have all jobs processed with the minimum *total completion time*, or *total flow time*, which is the summation of the duration each job has to wait until it is finished. Observe that if the assignment is known, the order each machine processes its assigned jobs is clear: It processes jobs in an increasing order of the processing time.

To be precise, the semi-matching problem is as follows. Let $G = (U \cup V, E)$ be a weighted bipartite graph, where U is a set of jobs and V is a set of machines. For any edge uv, let w_{uv} be its weight. Each weight of an edge uv indicates time it takes v to process u. Throughout this paper, let $n = |U \cup V|$, $m = |E|$. A set $M \subseteq E$ is a *semi-matching* if each job $u \in U$ is incident with exactly one edge in M. For any semi-matching M, we define the *cost* of M, denoted by $\mathrm{cost}(M)$, as follows. First, for any machine $v \in V$, its cost with respect to a semi-matching M is

$$\mathrm{cost}_M(v) = (w_1) + (w_1 + w_2) + \ldots + (w_1 + \ldots + w_{\deg_M(v)}) = \sum_{i=1}^{\deg_M(v)} (\deg_M(v) - i + 1) \cdot w_i$$

where $\deg_M(v)$ is the degree of v in M and $w_1 \leq w_2 \leq \ldots \leq w_{\deg_M(v)}$ are weights of the edges in M incident with v sorted increasingly. Intuitively, this is the total completion time of the jobs assigned to v. Note that for the unweighted case (i.e., when $w_e = 1$ for every edge e), the cost of a machine v is simply $\deg_M(v) \cdot (\deg_M(v) + 1)/2$. Now, the cost of the semi-matching M is simply the summation of the cost over all machines:

$$\mathrm{cost}(M) = \sum_{v \in V} \mathrm{cost}_M(v).$$

The goal is to find an *optimal semi-matching*, a semi-matching with minimum cost.

Previous work: Although the name "semi-matching" was recently proposed by Harvey, Ladner, Lovász, and Tamir [14], the problem was studied as early as 1970s when an $O(n^3)$ algorithm was independently developed by Horn [15] and Bruno et al. [5]. No progress has been made on this problem except on its special cases and variations. For the special case of *inclusive set restriction* where, for each pair of jobs u_1 and u_2, either all neighbors of u_1 are neighbors of u_2 or vice versa, a faster algorithm with $O(n^2)$ running time was given by Spyropoulos and Evans [31]. Many variations of this problem were recently proved to be NP-hard, including the preemtive version [30], the case when there are deadlines [32], and the case of optimizing total weighted tardiness [23]. The variation where the objective is to minimize $\max_{v \in V} \mathrm{cost}_M(v)$ was also considered [26,19].

The unweighted case of the semi-matching problem also received considerably attention in the past few years. Since it is shown by [14] that an optimal solution of the semi-matching problem is also optimal for the makespan version of the scheduling problem (where one wants to minimize the time the last machine finishes), we mention the results of both problems. The problem was first studied in a special case, called *nested* case where, for any two jobs, if their sets of

neighbors are not disjoint, then one of these sets contains the other set. This case is shown to be solvable in $O(m + n \log n)$ time [28, p.103]. For the general unweighted semi-matching problem, Abraham [1, Section 4.3] and Harvey et al. [14] independently develop two algorithms with $O(nm)$ running time. Lin and Li [22] also give an $O(n^3 \log n)$-time algorithm which is later generalized to a more general cost function [21]. Recently, [18] show that the problem can be solved in polynomial time even when there are release times.

The unweighted semi-matching problem is recently generalized to the quasi-matching problem by Bokal et al. [3]. In this problem, a function g is provided and each vertex $u \in U$ is required to connect to at least $g(u)$ vertices in v. Therefore, the semi-matching problem is when $g(u) = 1$ for every $u \in U$. They also develop an algorithm for this problem which is a generalization of the Hungarian method, and used it to deal with a routing problem in CDMA-based wireless sensor networks.

Motivated by the problem of assigning wireless stations (users) to access points, the unweighted semi-matching problem is also generalized to the problem of finding optimal semi-matching with minimum weight where an $O(n^2m)$ time algorithm is given [11].

Approximation algorithms and online algorithms for this problem (both weighted and unweighted cases) and the makespan version have also gained a lot of attention over the past few decades and have applications ranging from scheduling in hospital to wireless communication network. (See [20,39] for the recent surveys.)

Applications: As motivated by Harvey et al. [14], even in an online setting where the jobs arrive and depart over time, they may be reassigned from one machine to another cheaply if the algorithm's runtime is significantly faster than the arrival/departure rate. (One example of such case is the Microsoft Active Directory system [10,14].) The problem also arose from the Video on Demand (VoD) systems where the load of video disks needs to be balanced while data blocks from the disks are retrieved or while serving clients [25,36]. The problem, if solved in the distributed setting, can be used to construct a load balanced data gathering tree in sensor networks [29,27]. The same problem also arose in peer-to-peer systems [33,17,34].

In this paper, we also consider an "edge cover" version of the problem. In some applications such as sensor networks, there are no jobs and machines but the sensor nodes have to be clustered and each cluster has to pick its own head node to gather information from other nodes in the cluster. Motivated by this, Harada et al. [12] introduced the *balanced edge cover* problem[1] where the goal is to find an edge cover (set of edges incident to every vertex) that minimizes the total cost over all vertices. (The cost on each vertex is as previously defined.) They gave an $O(nm)$ algorithm for this problem and claimed that it could be used to solve the semi-matching problem as well. We show that this problem can be efficiently reduced to the semi-matching problem and thus our algorithm (on unweighted case) gives a better bound on this problem as well.

[1] This problem is also known as a *constant jump system* (see, e.g., [35,24]).

Our Results and Techniques

We consider the semi-matching problem and give a faster algorithm for each of the weighted and unweighted cases. We also extend the algorithm for the unweighted case to solve the balanced edge cover problem.

- **Weighted Semi-Matching:** (Section 2) We present an $O(nm \log n)$ algorithm, improving the previous $O(n^3)$ algorithm by Horn [15] and Bruno et al. [5]. As in the previous results [15,4,13], we use the reduction of the problem to the weighted bipartite matching problem as a starting point. We, however, only use the structural properties arising from the reduction and do not actually perform the reduction.
- **Unweighted Semi-Matching:** (Section 3) We give an $O(\sqrt{n}m \log n)$ algorithm, improving the previous $O(nm)$ algorithms by Abraham [1] and Harvey et al. [14].[2] Our algorithm uses the same reduction to the min-cost flow problem as in [14]. However, instead of canceling one negative cycle in each iteration, our algorithm exploits the structure of the graphs and the cost functions to cancel many negative cycles in a single iteration. This technique can also be generalized to other cost functions.
- **Balanced Edge Cover:** We also present a reduction from the balanced edge cover problem to the unweighted semi-matching problem. This leads to an $O(\sqrt{n}m \log n)$ algorithm for the problem, improving the previous $O(nm)$ algorithm by Harada et al. [12]. The main idea is to identify the "center" vertices of all the clusters in the optimal solution. (Note that any balanced edge cover (in fact, any minimal edge cover) clusters the vertices into stars.) Then, we partition the vertices into two sides, center and non-center ones, and apply the semi-matching algorithm on this graph. This result can be found in the full version ([9]).

Due to space limitation, most proofs are omitted and can be found in the full version [9].

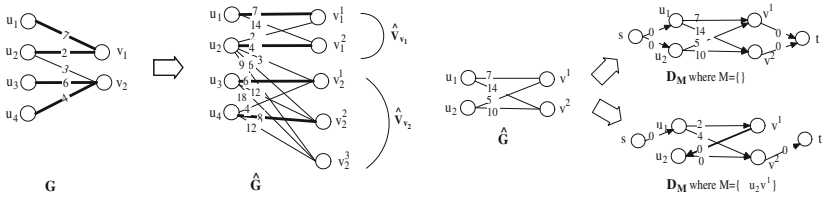
2 Weighted Semi-matching

In this section, we present an algorithm that finds optimal weighted semi-matching in $O(nm \log n)$ time.

Overview: Our improvement follows from studying the reduction from the weighted semi-matching problem to the weighted bipartite matching problem considered in the previous works [15,5,13] and the Edmonds-Karp-Tomizawa (EKT) algorithm for finding the weighted bipartite matching [8,38] . We first review these briefly.

Reduction: As in [15,5,13], we consider the reduction from the semi-matching problem on bipartite graph $G = (U \cup V, E)$ to the minimum-weight bipartite

[2] We also observe an $O(n^{5/2} \log n)$ algorithm that arises directly from the reduction by applying [16].



(a) Reduction (b) Residual graphs

Fig. 1.

matching on a graph $\hat{G}$. (See Figure 1(a).) The reduction is done by exploding the vertices in V, i.e, from a vertex v we create $\deg(v)$ vertices, $v^1, v^2, \ldots, v^{\deg(v)}$. For each edge incident to v^i in $\hat{G}$, we set its weight to i times its original weight in G. Denote the set of these vertices by $\hat{V}_v$. The correctness of this reduction can be seen by replacing the edges incident to v in the semi-matching by the edges incident to $v^1, v^2, \ldots$ with weights in an decreasing order. For example, in Figure 1(a), edge u_1v_1 and edge u_2v_1 in the semi-matching in G correspond to $u_1v_1^1$ and $u_2v_1^2$ in the matching in $\hat{G}$.

EKT algorithm: Our improvement comes from studying the behavior of the EKT algorithm for finding the bipartite matching in $\hat{G}$. The EKT algorithm iteratively increases the cardinality of the matching by one by finding a shortest augmenting path. Such path can be found by applying Dijkstra's algorithm on the *residual graph* D_M (corresponding to a matching M) with a *reduced cost*, denoted by $\tilde{w}$ as an edge length.

Figure 1(b) shows examples of residual graph D_M. Direction of an edge depends on whether it is in the matching or not. The weight of each edge depends on its weight in the original graph and the costs on its end vertices. We draw an edge of length 0 from s to all vertices in U_M and from all vertices in $\hat{V}_M$ to t, where U_M and $\hat{V}_M$ are the sets of unmatched vertices in U and $\hat{V}$, respectively. We want to find the shortest path from s to t or, equivalently, from U_M to $\hat{V}_M$.

The reduced cost is computed from the *potentials* on the vertices, which can be found as in Algorithm 1.[3]

Applying EKT algorithm directly leads to an $O(n(n' \log n' + m'))$ where $n = |U|$, $n' = |U \cup V|$ and m' is the number of edges in $\hat{G}$. Since $n' = |\hat{V}| = \Theta(m)$ and $m' = \Theta(n^2)$, the running time is $O(nm \log n + n^3)$. (We note that this could be brought down to $O(n^3)$ using Kao et al.'s trick [16] of reducing the number of participating edges.) The bottleneck here is the Dijkstra's algorithm which needs $O(n' \log n' + m')$ time. We now review this algorithm and pinpoint the part that will be sped up.

[3] We note that we set the potentials in an unusual way: We keep potentials of the unmatched vertices in $\hat{V}$ to 0. The reason is roughly that we can speed up the process of finding the distances of all vertices but vertices in $\hat{V}_M$. Notice that this type of potentials is valid too (i.e., $\tilde{w}$ is non-negative) since for any edge uv such that $v \in \hat{V}_M$ is unmatched, $\tilde{w}(uv) = w_{uv} + p(u) - p(v) = w_{uv} + p(u) \geq 0$.

Algorithm 1. EKT Algorithm $(\hat{G}, w)$

1. Let $M = \emptyset$.
2. For every node v, let $p(v) = 0$. ($p(v)$ is a potential on v.)
3. **repeat**
4. Let $\tilde{w}_{uv} = w_{uv} + p(u) - p(v)$ for every edge uv. ($\tilde{w}_{uv}$ is a reduced cost of an edge uv.)
5. For every node v, compute the distance $d(v)$ which is the distance from U_M (the set of unmatched vertices in U) to v in D_M. (Recall that the length of edges in D_M is $\tilde{w}$.)
6. Let P be the shortest U_M-$\hat{V}_M$ path in D_M.
7. Update the potential $p(u)$ to $d(u)$ for every vertex $u \in U \cup (\hat{V} \setminus \hat{V}_M)$.
8. Augment M along P, i.e., $M = P \triangle M$ (where $\triangle$ denotes the symmetric difference operator).
9. **until** all vertices in U are matched
10. **return** M

Dijkstra's algorithm: Recall that the Dijkstra's algorithm starts from a source vertex and keeps adding to its shortest path tree a vertex with minimum tentative distance. When a new vertex v is added, the algorithm updates the tentative distance of all vertices outside the tree by relaxing *all* edges incident to v. On an n'-vertex m'-edge graph, it takes $O(\log n')$ time (using priority queue) to find a new vertex to add to the tree and hence $O(n' \log n')$ in total. Further, relaxing all edges takes $O(m')$ time in total. Recall that in our case, $m' = \Theta(n^2)$ which is too large. *Thus, we wish to reduce the number of edge relaxations to improve the overall running time.*

Our approach: We reduce the number of edge relaxation as follows. Suppose that a vertex $u \in U$ is added to the shortest path tree. For every $v \in V$, a neighbor of u in G, we relax all edges uv^1, uv^2, ..., uv^i in $\hat{G}$ *at the same time.* In other words, instead of relaxing $\Theta(nm)$ edges in $\hat{G}$ separately, we group the edges to m groups (according to the edges in G) and relax all edges in each group together. We develop a relaxation method that takes $O(\log n)$ time per group. In particular, we design a data structure H_v, for each vertex $v \in V$, that supports the following operations.

- Relax(uv, H_v): This operation works as if it relaxes edges uv^1, uv^2, ...
- AccessMin(H_v): This operation returns a vertex v^i (exploded from v) with minimum tentative distance among vertices that are not deleted (by the next operation).
- DeleteMin(H_v):This operation finds v^i from AccessMin then returns and deletes v^i.

Our main result is that, by exploiting the structure of the problem, one can design H_v that supports Relax, AccessMin and DeleteMin in $O(\log n)$, $O(1)$ and $O(\log n)$, respectively. Before showing such result, we note that speeding up Dijkstra's algorithm and hence EKT algorithm is quite straightforward once we have H_v: We simply build a binary heap H whose nodes correspond to vertices

in an original graph G. For each vertex $u \in U$, H keeps track of its tentative distance. For each vertex $v \in V$, H keeps track of its *minimum tentative distance* returned from H_v.

Main idea: Before going into details, we sketch the main idea here. The data structure H_v that allows fast "group relaxation" operation can be built because of the following nice structure of the reduction: For each edge uv of weight w_{uv} in G, the weights $w_{uv^1}, w_{uv^2}, \ldots$ of the corresponding edges in $\hat{G}$ increase linearly (i.e., $w_{uv}, 2w_{uv}, 3w_{uv}, \ldots$). This enables us to know the order of vertices, among $v^1, v^2, \ldots$, that will be added to the shortest path tree. For example, in Figure 1(b), when $M = \emptyset$, we know that, among v^1 and v^2, v^1 will be added to the shortest path tree first as it always has a smaller tentative distance.

However, since the length of edges in D_M does not solely depend on the weights of the edges in $\hat{G}$ (in particular, it also depends on a potentials on both end vertices), it is possible (after some iterations of the EKT algorithm) that v^1 is added to the shortest path tree after v^2.

Fortunately, due to the way the potential is defined by the EKT algorithm, a similar nice property still holds: Among $v^1, v^2, \ldots$ in D_M corresponding to v in G, if a vertex v^k, for some k, is added to the shortest path tree first, then the vertices on each side of v^k have a nice order: Among $v^1, v^2, \ldots, v^{k-1}$, the order of vertices added to the shortest path tree is $v^{k-1}, v^{k-2}, \ldots, v^2, v^1$. Further, among $v^{k+1}, v^{k+2}, \ldots$, the order of vertices added to the shortest path tree is $v^{k+1}, v^{k+2}, \ldots$.

This main property, along with a few other observations, allow us to construct the data structure H_v. We now show the properties we need.

Properties of the Tentative Distance

Consider any iteration of the EKT algorithm (with a potential function p and a matching M). We study the following functions f_{*_v} and g_{*v}.

Definition 1. *For any edge uv from U to V and any integer $1 \le i \le \deg(v)$, let*

$$g_{uv}(i) = d(u) + p(u) + i \cdot w_{uv} \quad \text{and} \quad f_{uv}(i) = g_{uv}(i) - p(v^i) = d(u) + p(u) - p(v^i) + i \cdot w_{uv}.$$

For any $v \in V$ and $i \in [1, \deg(v)]$, define the lower envelope of f_{uv} and g_{uv} over all $u \in U$ as

$$f_{*v}(i) = \min_{u:(u,v)\in E} f_{uv}(i) \qquad \text{and} \quad g_{*v}(i) = \min_{u:(u,v)\in E} g_{uv}(i).$$

Our goal is to understand the structure of the function f_{*v} which is the tentative distance of $v^1, v^2, \ldots$. The function g_{*v} is simply f_{*v} with the potential of v ignored. We define g_{*v} as it is easier to keep track of (since it is piecewise linear, as in Proposition 1). Now we state the key properties that enable us to keep track of f_{*v} efficiently. Recall that $v^1, v^2, \ldots$ are the exploded vertices of v (from the reduction).

Proposition 1. *Consider a matching M and a potential p at any iteration of the EKT algorithm.*

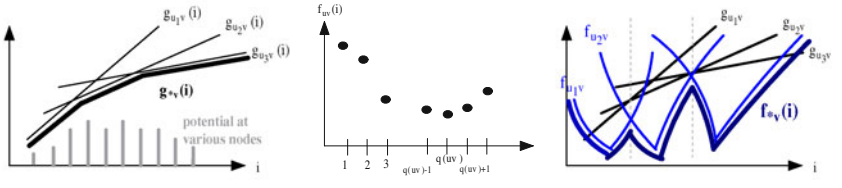
(1) For any vertex $v \in V$, there exists α_v such that $v^1, \ldots, v^{\alpha_v}$ are all matched and $v^{\alpha_v+1}, \ldots, v^{\deg(v)}$ are all unmatched.

(2) For any vertex $v \in V$, g_{*v} is a piecewise linear function.

(3) For any edge $uv \in E$ where $u \in U$ and $v \in V$, and any i, $f_{uv}(i) = f_{*v}(i)$ if and only if $g_{uv}(i) = g_{*v}(i)$.

(4) For any edge $uv \in E$ where $u \in U$ and $v \in V$, let α_v be as in (1). There exists an integer $1 \leq \gamma_{uv} \leq k$ such that for $i = 1, 2, \ldots, \gamma_{uv} - 1$, $f_{uv}(i) > f_{uv}(i+1)$ and for $i = \gamma_{uv}, \gamma_{uv} + 1, \ldots, \alpha_v - 1$, $f_{uv}(i) \leq f_{uv}(i+1)$. In other words, $f_{uv}(1), f_{uv}(2), \ldots, f_{uv}(\alpha_v)$ is a unimodal sequence.

Figure 2(a) and 2(b) show the structure of g_{*v} and f_{*v} according to statement (2) and (4) in the above proposition. By statement (3), the two pictures can be combined as in Figure 2(c): g_{*v} indicates u that makes both g_{*v} and f_{*v} minimum in each interval and one can find i that minimizes f_{*v} in each interval by looking at α_v (or near α_v in some case).



(a) g_{*v} and potential function. Note that $w_{u_1v} > w_{u_2v} > w_{u_3v}$.

(b) f_{*v} is unimodal.

(c) f_{*v} together with g_{*v}.

Fig. 2.

The key idea in constructing the data structure is that, for each edge $uv \in E$, one can maintain the minimum tentative distance from u to $v^1, v^2, \ldots, v^\alpha$ by simply keeping two pointers (starting from α_v and moving left and right). This is because $f_{uv}(1), f_{uv}(2), \ldots, f_{uv}(\alpha_v)$ is unimodal, as previously shown. Thus, we only need to find v^i from edges $e \in E$ incident to v. Details can be found in the full version.

3 Unweighted Semi-matching

In this section, we present an algorithm that finds the optimal semi-matching in unweighted graph in $O(m\sqrt{n}\log n)$ time.

Overview: Our algorithm consists of the following three steps.

In the first step, we reduce the problem on graph G to the min-cost flow problem on network N, using the same reduction from Harvey et al. [14]. (See

Figure 3. Details are in the full version.) We note that the flow is optimal if and only if there is no cost reducing path (to be defined later). We start with an arbitrary semi-matching and use this reduction to get a corresponding flow. The goal is to eliminate all the cost-reducing paths.

The second step is a divide-and-conquer algorithm used to eliminate all the cost-reducing paths. We call this algorithm CancelAll (cf. Algorithm 2). The main idea here is that the vertices can be divided into two sets so that eliminating cost reducing paths "inside" each set does not introduce any new cost reducing paths anywhere in the graph. This dividing step needs to be done carefully. We treat this in Section 3.1.

Finally, in the last component of the algorithm we deal with eliminating cost-reducing paths between two sets of vertices quickly. Naively, one can do this using any unit-capacity max-flow algorithm. To get a faster algorithm, we observe that the structure of the graph is similar to a *unit network*, where every vertex has in-degree or out-degree one. Thus, we get the same performance guarantee as the Dinitz's algorithm [6,7]. [4] Details of this part can be found in Section 3.2.

Due to space limitation, details on running time and generalization are omitted here and can be found in the full version.

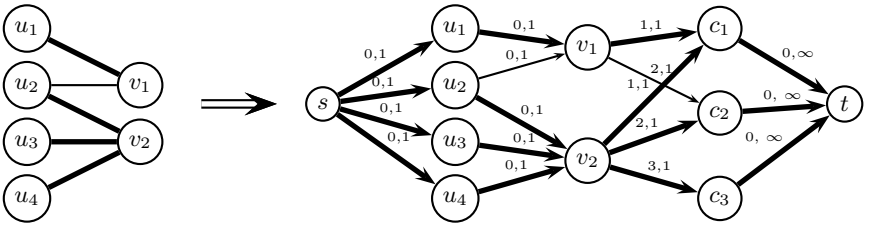


Fig. 3. Reduction to the min-cost flow problem. Each edge is labeled with **(cost, capacity)** constraint. Thick edges either are matching edges or contain the flow.

3.1 The Divide-and-Conquer Algorithm

Our algorithm takes a bipartite graph $G = (U \cup V, E')$ and outputs the optimal semi-matching. It starts by transforming G into a graph N as described in the previous section. Since the source s and the sink t are always clear from the context, the graph N can be seen as a tripartite graph with vertices $U \cup V \cup C$; later on we denote $N = (U \cup V \cup C, E)$. The algorithm proceeds by finding an arbitrary max-flow f from s to t in N which corresponds to a semi-matching in G. This can be done in linear time since the flow is equivalent to any semi-matching in G.

To find the min-cost flow in N, the algorithm uses a subroutine called CancelAll (cf. Algorithm 2) to cancel all cost-reducing paths in f.

[4] The algorithm is also known as "Dinic's algorithm". See [7] for details.

Algorithm 2. CancelAll($N = (U \cup V \cup C, E)$)

1. **if** $|C| = 1$ **then** halt **endif**
2. Divide C into C_1 and C_2 of roughly equal size.
3. Cancel(N, C_2, C_1). {Cancel all cost-reducing paths from C_2 to C_1}.
4. Divide N into N_1 and N_2 where N_2 is "reachable" from C_2 and N_1 is the rest.
5. Recursively solve CancelAll(N_1) and CancelAll(N_2).

CancelAll works by dividing C and solves the problem recursively. Given a set of cost centers C, the algorithm divides C into roughly equal-size subsets C_1 and C_2 such that, for any $c_i \in C_1$ and $c_j \in C_2$, $i < j$. This guarantees that there is no cost reducing path from C_1 to C_2. Then it cancels all cost reducing paths from C_2 to C_1 by calling Cancel algorithm (described in Section 3.2).

It is left to cancel the cost-reducing paths "inside" each of C_1 and C_2. This is done by partitioning the vertices of N (except s and t) into two graphs N_1 and N_2 and solve the problem separately on each of them. The partition is done by letting N_2 be a subgraph induced by vertices reachable from C_2 in the residual graph and N_1 be the subgraph induced by the rest vertices. (Note that both graphs have s and t.) For example, in Figure 3, v_1 is reachable from c_3 by the path c_3, v_2, u_2, v_1 in the residual graph.

Lemma 1. CancelAll(N) *(cf. Algortihm 2) cancels all cost-reducing paths in* N.

3.2 Canceling Paths from C_2 to C_1

We cancels all admissible paths from C_2 to C_1 in R_f by running Dinitz's blocking flow algorithm [6] from a super-source s connecting to vertices in C_2 to a super-sink t connecting to vertices in C_1. We exploit the properties that N is unit-capacity and every vertex of U has indegree 1 in R_f to show (by a proof similar to that in [37, Theorem 8.8]) that this algorithm runs in $O(|E|\sqrt{|U|})$ time. Details of this algorithm and the total running time can be found in the full version.

Acknowledgment. We thank David Pritchard for useful suggestion and Jane (Pu) Gao for pointing out some related surveys.

References

1. Abraham, D.: Algorithmics of two-sided matching problems. Master's thesis, Department of Computer Science, University of Glasgow (2003)
2. Blażewicz, J., Ecker, K., Pesch, E., Schmidt, G., Weglarz, J.: Handbook on scheduling: from theory to applications
3. Bokal, D., Bresar, B., Jerebic, J.: A generalization of hungarian method and hall's theorem with applications in wireless sensor networks. arXiv/0911.1269 (2009)
4. Bruno, J.L., Coffman Jr., E.G., Sethi, R.: Algorithms for minimizing mean flow time. In: IFIP Congress, pp. 504–510 (1974)

5. Bruno, J.L., Coffman Jr., E.G., Sethi, R.: Scheduling independent tasks to reduce mean finishing time. ACM Commun. 17(7), 382–387 (1974)
6. Dinic, E.A.: Algorithm for solution of a problem of maximum flow in networks with power estimation. Soviet Mathematics Doklady 11, 1277–1280 (1970)
7. Dinitz, Y.: Dinitz' algorithm: The original version and even's version. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds.) Theoretical Computer Science. LNCS, vol. 3895, pp. 218–240. Springer, Heidelberg (2006)
8. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. In: Combinatorical Structures and Their Applications, pp. 93–96. Gordon and Breach, New York (1970)
9. Fakcharoenphol, J., Lekhanukit, B., Nanongkai, D.: Faster algorithms for semi-matching problems. arXiv/1004.3363 (2010)
10. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5(2), 287–326 (1979)
11. Harada, Y., Ono, H., Sadakane, K., Yamashita, M.: Optimal balanced semi-matchings for weighted bipartite graphs. IPSJ Digital Courier 3, 693–702 (2007)
12. Harada, Y., Ono, H., Sadakane, K., Yamashita, M.: The balanced edge cover problem. In: Hong, S.-H., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 246–257. Springer, Heidelberg (2008)
13. Harvey, N.J.A.: Semi-matchings for bipartite graphs and load balancing (slide) (July 2003), http://people.csail.mit.edu/nickh/Publications/SemiMatching/Semi-Matching.ppt
14. Harvey, N.J.A., Ladner, R.E., Lovász, L., Tamir, T.: Semi-matchings for bipartite graphs and load balancing. In: Dehne, F., Sack, J.-R., Smid, M. (eds.) WADS 2003. LNCS, vol. 2748, pp. 53–78. Springer, Heidelberg (2003)
15. Horn, W.A.: Minimizing average flow time with parallel machines. Operations Research, 846–847 (1973)
16. Kao, M.-Y., Lam, T.W., Sung, W.-K., Ting, H.-F.: An even faster and more unifying algorithm for comparing trees via unbalanced bipartite matchings. J. Algorithms 40(2), 212–233 (2001)
17. Kothari, A., Suri, S., Tóth, C.D., Zhou, Y.: Congestion games, load balancing, and price of anarchy. In: López-Ortiz, A., Hamel, A.M. (eds.) CAAN 2004. LNCS, vol. 3405, pp. 13–27. Springer, Heidelberg (2005)
18. Lee, K., Leung, J.Y.-T., Pinedo, M.L.: Scheduling jobs with equal processing times subject to machine eligibility constraints. Working Paper (2009)
19. Lee, K., Leung, J.Y.T., Pinedo, M.L.: A note on an approximation algorithm for the load-balanced semi-matching problem in weighted bipartite graphs. Inf. Process. Lett. 109(12), 608–610 (2009)
20. Leung, J.Y.-T., Li, C.-L.: Scheduling with processing set restrictions: A survey. International Journal of Production Economics 116(2), 251–262 (2008)
21. Li, C.-L.: Scheduling unit-length jobs with machine eligibility restrictions. European Journal of Operational Research 174(2), 1325–1328 (2006)
22. Lin, Y., Li, W.: Parallel machine scheduling of machine-dependent jobs with unit-length. European Journal of Operational Research 156(1), 261–266 (2004)
23. Logendran, R., Subur, F.: Unrelated parallel machine scheduling with job splitting. IIE Transactions 36(4), 359–372 (2004)
24. Lovász, L.: The membership problem in jump systems. J. Comb. Theory, Ser. B 70(1), 45–66 (1997)

25. Low, C.P.: An efficient retrieval selection algorithm for video servers with random duplicated assignment storage technique. Inf. Process. Lett. 83(6), 315–321 (2002)
26. Low, C.P.: An approximation algorithm for the load-balanced semi-matching problem in weighted bipartite graphs. Inf. Process. Lett. 100(4), 154–161 (2006) (also appeared in TAMC 2006)
27. Machado, R., Tekinay, S.: A survey of game-theoretic approaches in wireless sensor networks. Computer Networks 52(16), 3047–3061 (2008)
28. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems, 2nd edn. Prentice Hall, Englewood Cliffs (2001)
29. Sadagopan, N., Singh, M., Krishnamachari, B.: Decentralized utility-based sensor network design. Mob. Netw. Appl. 11(3), 341–350 (2006)
30. Sitters, R.: Two NP-hardness results for preemptive minsum scheduling of unrelated parallel machines. In: Aardal, K., Gerards, B. (eds.) IPCO 2001. LNCS, vol. 2081, pp. 396–405. Springer, Heidelberg (2001)
31. Spyropoulos, C.D., Evans, D.J.: Analysis of the q.a.d. algorithm for an homogeneous multiprocessor computing model with independent memories. International Journal of Computer Mathematics, 237–255
32. Su, L.H.: Scheduling on identical parallel machines to minimize total completion time with deadline and machine eligibility constraints. The International Journal of Advanced Manufacturing Technology 40(5), 572–581 (2009)
33. Suri, S., Tóth, C.D., Zhou, Y.: Uncoordinated load balancing and congestion games in p2p systems. In: IPTPS, pp. 123–130 (2004)
34. Suri, S., Tóth, C.D., Zhou, Y.: Selfish load balancing and atomic congestion games. Algorithmica 47(1), 79–96 (2007) (also appeared in SPAA 2004)
35. Tamir, A.: Least majorized elements and generalized polymatroids. Mathematics of Operations Research, 583–589 (1995)
36. Tamir, T., Vaksendiser, B.: Algorithms for storage allocation based on client preferences. In: International Symposium on Combinatorial Optimization (2008)
37. Tarjan, R.E.: Data structures and network algorithms. Society for Industrial and Applied Mathematics (1983)
38. Tomizawa, N.: On some techniques useful for solution of transportation network problems. In: Networks 1, pp. 173–194 (1971)
39. Vaik, Z.: On scheduling problems with parallel multi-purpose machines. Technical report, Technical Reports, Egervary Research Group on Combinatorial Optimization, Hungary (2005), http://www.cs.elte.hu/egres/tr/egres-05-02.pdf

Clustering with Diversity

Jian Li[1], Ke Yi[2], and Qin Zhang[2]

[1] University of Maryland, College Park, MD, USA
lijian@cs.umd.edu
[2] Hong Kong University of Science and Technology, Hong Kong, China
{yike,qinzhang}@cse.ust.hk

Abstract. We consider the *clustering with diversity* problem: given a set of colored points in a metric space, partition them into clusters such that each cluster has at least ℓ points, all of which have distinct colors. We give a 2-approximation to this problem for any ℓ when the objective is to minimize the maximum radius of any cluster. We show that the approximation ratio is optimal unless $\mathbf{P} = \mathbf{NP}$, by providing a matching lower bound. Several extensions to our algorithm have also been developed for handling outliers. This problem is mainly motivated by applications in privacy-preserving data publication.

Keywords: Approximation algorithm, k-center, k-anonymity, l-diversity.

1 Introduction

Clustering is a fundamental problem with a long history and a rick collection of results. A general clustering problem can be formulated as follows. Given a set of points P in a metric space, partition P into a set of disjoint clusters such that a certain objective function is minimized, subject to some cluster-level and/or instance-level constraints. Typically, cluster-level constraints impose restrictions on the number of clusters or on the size of each cluster. The former corresponds to the classical k-center, k-median, k-means problems, while the latter has recently received much attention from various research communities [1, 15, 16]. On the other hand, instance-level constraints specify whether particular items are similar or dissimilar, usually based on some background knowledge [3, 26]. In this paper, we impose a natural instance-level constraint on a clustering problem, that the points are colored and all points partitioned into one cluster must have distinct colors. We call such a problem *clustering with diversity*. Note that the traditional clustering problem is a special case of ours where all points have unique colors.

As an illustrating example, consider the problem of choosing locations for a number of factories in an area where different resources are scattered. Each factory needs at least ℓ different resources allocated to it and the resource in one location can be sent to only one factory. This problem corresponds to our clustering problem where each kind of resource has a distinct color, and we have a lower bound ℓ on the the cluster size.

The main motivation to study clustering with diversity is privacy preservation for data publication, which has drawn tremendous attention in recent years in both the database community [7, 8, 21, 24, 28–30] and the theory community [1, 2, 11, 12, 22].

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 188–200, 2010.

The goal of all the studies in privacy preservation is to prevent *linking attacks* [25]. Consider the table of patient records in Figure 1(a), usually called the *microdata*. There are three types of attributes in a microdata table. The *sensitive attribute* (SA), such as "Disease", is regarded as the individuals' privacy, and is the target of protection. The *identifier*, in this case "Name", uniquely identifies a record, hence must be ripped off before publishing the data. The rest of the attributes, such as "Age", "Gender", and "Education", should be published so that researchers can apply data mining techniques to study the correlation between these attributes and "Disease". However, since these attributes are public knowledge, they can often uniquely identify individuals when combined together. For example, if an attacker knows (i) the age (25), gender (M), and education level (Master) of Bob, and (ii) Bob has a record in the microdata, s/he easily finds out that Tuple 2 is Bob's record and hence, Bob contracted HIV. Therefore, these attributes are often referred to as the *quasi-identifiers* (QI). The solution is thus to make these QIs ambiguous before publishing the data so that it is difficult for an attacker to link an individual from the QIs to his/her SA, but at the same time we want to minimize the amount of information loss due to the ambiguity introduced to the QIs so that the interesting correlations between the QIs and the SA are still preserved.

T-1D(*Name*)	Age	Gender	Degree	Disease
1 (*Adam*)	29	M	M.Sc.	HIV
2 (*Bob*)	25	M	M.Sc.	HIV
3 (*Calvin*)	25	M	B.Sc.	pneumonia
4 (*Daisy*)	29	F	B.Sc.	bronchitis
5 (*Elam*)	40	M	B.Sc.	bronchitis
6 (*Frank*)	45	M	B.Sc.	bronchitis
7 (*George*)	35	M	B.Sc.	pneumonia
8 (*Henry*)	37	M	B.Sc.	pneumonia
9 (*Ivy*)	50	F	Ph.D.	dyspepsia
10 (*Jane*)	60	F	Ph.D.	pneumonia

(a)

QIs	Disease
(center, radius)	HIV
	HIV
(center, radius)	pneumonia
	bronchitis
(center, radius)	bronchitis
	bronchitis
(center, radius)	pneumonia
	pneumonia
(center, radius)	dyspepsia
	pneumonia

(b)

QIs	Disease
(center, radius)	HIV
	HIV
	pneumonia
	bronchitis
(center, radius)	bronchitis
	bronchitis
	pneumonia
	pneumonia
(center, radius)	dyspepsia
	pneumonia

(c)

Fig. 1. (a) The microdata; (b) A 2-anonymous table; (c) An 2-diverse table

The usual approach taken to prevent linking attacks is to partition the tuples into a number of *QI-groups*, namely clusters, and within each cluster all the tuples share the same (ambiguous) QIs. There are various ways to introduce ambiguity. A popular approach, as taken by [1], is to treat each tuple as a high-dimensional point in the QI-space, and then only publish the center, the radius, and the number of points of each cluster. To ensure a certain level of privacy, each cluster is required to have at least k points so that the attacker is not able to correctly identify an individual with confidence larger than $1/k$. This requirement is referred to as the k-*ANONYMITY* principle [1, 22]. The problem, translated to a clustering problem, can be phrased as follows: Cluster a set of points in a metric space, such that each cluster has at least r points. When the objective is to minimize the maximum radius of all clusters, the problem is called r-GATHERING and a 2-approximation is known [1].

However, the k-ANONYMITY principle suffers from the *homogeneity* problem: A cluster may have too many tuples with the same SA value. For example, in Figure 1(b),

all tuples in QI-group 1, 3, and 4 respectively have the the same disease. Thus, the attacker can infer what disease all the people within a QI-group have contracted without identifying any individual record. The above problem has led to the development of many SA-aware principles. Among them, ℓ-*DIVERSITY* [21] is the most widely deployed [8, 13, 17, 21, 28, 29], due to its simplicity and good privacy guarantee. The principle demands that, in each cluster, at most $1/\ell$ of its tuples can have the same SA value. Figure 1(c) shows a 2-diverse version of the microdata. In an ℓ-diverse table, an attacker can figure out the real SA value of the individual with confidence no more than $1/\ell$. Treating the SA values as colors, this problem then exactly corresponds to the clustering with diversity problem defined at the beginning, where we have a lower bound ℓ on the cluster size.

In contrast to the many theoretical results for r-GATHERING and k-ANONYMITY [1, 2, 22], no approximation algorithm with performance guarantees is known for ℓ-DIVERSITY, even though many heuristic solutions have been proposed [13, 19, 21].

Clustering with instance-level constraints and other related work. Clustering with instance-level constraints is a developing area and begins to find many interesting applications in various areas such as bioinformatics [5], machine learning [26, 27], data cleaning [4], etc. Wagstaff and Cardie in their seminal work [26] considered the following two types of instance-level hard constraints: A *must-link (ML)* constraint dictates that two particular points must be clustered together and a *cannot-link (CL)* constraint requires they must be separated. Many heuristics and variants have been developed subsequently, e.g. [27, 31], and some hardness results with respect to minimizing the number of clusters were also obtained [10]. However, to the best of our knowledge, no approximation algorithm with performance guarantee is known for any version of the problem. We note that an ℓ-diverse clustering can be seen as a special case where nodes with the same color must satisfy CL constraints.

As opposed to the hard constraints imposed on any clustering, the correlation clustering problem [6] considers soft and possibly conflicting constraints and aims at minimizing the violation of the given constraints. An instance of this problem can be represented by a complete graph with each edge labeled (+) or (-) for each pair of vertices, indicating that two vertices should be in the same or different clusters, respectively. The goal is to cluster the elements so as to minimize the number of disagreements, i.e., (-) edges within clusters and (+) edges crossing clusters. The best known approximations for various versions of the problem are due to Ailon et al. [3]. If the number of clusters is stipulated to be a small constant k, there is a polynomial time approximation scheme [14]. In the Dedupalog project, Arasu et al. [4] considered correlation clustering together with instance-level hard constraints, with the aim of de-duplicating entity references .

Approximation algorithms for clustering with outliers were first considered by Charikar et al. [9]. The best known approximation factor for r-GATHERING with outliers is 4 due to Aggrawal et al. [1].

Our results. In this paper, we give the first approximation algorithms to the clustering with diversity problem. We formally define the problem as follows.

Definition 1 (ℓ-DIVERSITY). *Given a set of n points in a metric space where each of them has a color, cluster them into a set $\mathcal{C}$ of clusters, such that each cluster has at least ℓ points, and all of its points have distinct colors. The goal is to minimize the maximum radius of any cluster.*

Our first result (Section 2) is a 2-approximation algorithm for ℓ-DIVERSITY. The algorithm follows a similar framework as in [1], but it is substantially more complicated. The difficulty is mainly due to the requirement to resolve the conflicting colors in each cluster while maintaining its minimum size ℓ. To the best of our knowledge, this is first approximation algorithm for a clustering problem with instance-level hard constraints.

Next, we show that this approximation ratio is the best possible by presenting a matching lower bound (Section 3). A lower bound of 2 is also given in [1] for r-GATHERING. But to carry that result over to ℓ-DIVERSITY, all the points need to have unique colors. This severely limits to applicability of this hardness result. In Section 3 we give a construction showing that even with only 3 colors, the problem is NP-hard to approximate within any factor strictly less than 2. In fact, if there are only 2 colors, we show that the problem can be solved optimally in polynomial time via bipartite matching.

Unlike r-GATHERING, an instance to the ℓ-DIVERSITY problem may not have a feasible solution at all, depending on the color distribution. In particular, we can easily see that no feasible clustering exists when there is one color that has more than $\lfloor n/\ell \rfloor$ points. One way to get around this problem is to have some points not clustered (which corresponds to deleting a few records in the ℓ-DIVERSITY problem). Deleting records causes information loss in the published data, hence should be minimized. Ideally, we would like to delete points just enough such that the remaining points admit a feasible ℓ-diverse clustering. In Section 4, we consider the *ℓ-DIVERSITY-OUTLIERS* problem, where we compute an ℓ-diverse clustering after removing the least possible number of points. We give an $O(1)$-approximation algorithm to this problem.

Our techniques for dealing with diversity and cluster size constraints may be useful in developing approximation algorithms for clustering with more general instance-level constraints. Due to space constraints, we only provide complete details for our first result. We refer interested readers to the full version of the paper for all missing details and proofs [20].

2 A 2-Approximation for ℓ-DIVERSITY

In this section we assume that a feasible solution on a given input always exists. We first introduce a few notations. Given a set of n points in a metric space, we construct a weighted graph $G(V, E)$ where V is the set of points and each vertex $v \in V$ has a color $c(v)$. For each pair of vertices $u, v \in V$ with different colors, we have an edge $e = (u, v)$, and its weight $w(e)$ is just their distance in the metric space. For any $u, v \in V$, let $\texttt{dist}_G(u, v)$ be the shortest path distance of u, v in graph G. For any set $A \subseteq V$, let $N_G(A)$ be the set of neighbors of A in G. For a pair of sets $A \subseteq V, B \subseteq V$, let $E_G(A; B) = \{(a, b) \mid a \in A, b \in B, (a, b) \in E(G)\}$. The diameter of a cluster C of nodes is defined to be $d(C) = \max_{u,v \in C}(w(e(u, v)))$. Given a cluster C and its center v, the radius $r(C)$ of C is defined as maximum distance from any node of

C to v, i.e., $r(C) = \max_{u \in C} w(u, v)$. By triangle inequality, it is obvious to see that $\frac{1}{2}d(C) \leq r(C) \leq d(C)$.

A *star forest* is a forest where each connected component is a star. A *spanning star forest* is a star forest spanning all vertices. The *cost* of a spanning forest $\mathcal{F}$ is the length of the longest edge in $\mathcal{F}$. We call a star forest *semi-valid* if each star component contains at least ℓ colors and *valid* if it is semi-valid and each star is *polychromatic*, i.e., each node in the star has a distinct color. Note that a spanning star forest with cost R naturally defines a clustering with largest radius R. Denote the radius and the diameter of the optimal clustering by r^* and d^*, respectively.

We first briefly review the 2-approximation algorithm for the r-GATHERING problem [1], which is the special case of our problem when all the points have distinct colors. Let $e_1, e_2, \ldots$ be the edges of G in a non-decreasing order of their weights. The general idea of the r-GATHERING algorithm [1] is to first guess the optimal radius R by considering each graph G_i formed by the first i edges $E_i = \{e_1, \ldots, e_i\}$, as $i = 1, 2, \ldots$. It is easy to see that the cost of a spanning star forest of G_i is at most $w(e_i)$. For each G_i $(1 \leq i \leq m)$, the following condition is tested (rephrased to fit into our context):

(I) There exists a maximal independent set I such that there is a spanning star forest in G_i with the nodes in I being the star centers, and each star has at least r nodes.

It is proved [1] that the condition is met if the length of e_i is d^*. The condition implies the radius of our solution is at most d^* which is at most $2r^*$. Therefore, we get an 2-approximation. In fact, the independent set I can be chosen greedily and finding the spanning star forest can be done via a network flow computation.

Our 2-approximation for the ℓ-diversity problem follows the same framework, that is, we check each G_i in order and test the following condition:

(II) There exists a maximal independent set I such that there is a *valid* spanning star forest in G_i with the nodes in I being the star centers.

The additional challenge is of course that, while condition (I) only puts a constraint on the size of each star, condition (II) requires both the size of each star to be at least ℓ and all the nodes in a star have distinct colors. Below we first give a constructive algorithm that for a given G_i, tries to find an I such that condition (II) is met. Next we show that when $w(e_i) = d^*$, the algorithm is guaranteed to succeed. The approximation ratio of 2 then follows immediately.

To find an I to meet condition (II), the algorithm starts with an arbitrary maximal independent set I, and iteratively augments it until the condition is met, or fails otherwise. In each iteration, we maintain two tests. The first one, denoted by *flow test* F-TEST(G_i, I), checks if there exists a semi-valid spanning star forest in G_i with nodes in I being star centers. If I does not pass this test, the algorithm fails right away. Otherwise we go on to the second test, denoted by *matching test* M-TEST(G_i, I), which tries to find a valid spanning star forest. If this test succeeds, we are done; otherwise the failure of this test yields a way to augment I and we proceed to the next iteration. The algorithm is outlined in Algorithm 1.

We now elaborate on F-TEST and M-TEST. F-TEST(G_i, I) checks if there is a spanning star forest in G_i with I being the star centers such that each star contains at least

Algorithm 1: Algorithm to find an I in G_i to meet condition (II)

```
Let I be an arbitrary maximal independent set in G_i;
while F-TEST(G_i, I) is passed do
    (S, S') ← M-TEST(G_i, I) /* S ⊂ V, S' ⊆ I */;
    if S = ∅ then
        Succeed;
    else
        I ← I − S' + S;
        Add nodes to I until it is a maximal independent set;
Fail;
```

ℓ colors. As the name suggests, we conduct the test using a network flow computation. We first create a source s and a sink t. For each node $v \in V$, we add an edge (s, v) with capacity 1, and for each node $o_j \in I (1 \leq j \leq |I|)$, we create a vertex o_j and add an outgoing edge (o_j, t) with capacity lower bound ℓ. For each node $o_j \in I$ and each color c, we create a vertex $p_{j,c}$ and an edge $(p_{j,c}, o_j)$ with capacity upper bound 1. For any $v \in V$ such that $(v, o_j) \in E_i$ or $v = o_j$, and v has color c, we add an edge from v to $p_{j,c}$ without capacity constraint. Finally, we add one new vertex o'_j for each $o_j \in I$, connect all $v \in V$ to o'_j without capacity constraint if $(v, o_j) \in E$ or $v = o_j$, and connect o'_j to t without capacity constraint. The capacity upper bound of $(p_{j,c}, o_j)$ forces at most one node with color c to be assigned to o_j. Therefore, all nodes assigned to o_j have distinct colors. The capacity lower bounds of (o_j, t)s require that each cluster has at least ℓ nodes. Nodes o'_js are used to absorb other unassigned nodes. It is not difficult to see that there exists a semi-valid spanning star forest with nodes in I being star centers in G_i if an n-units flow can be found. In this case we say that the F-TEST is passed. See Figure 2 for an example. Note that a network flow problem with both capacity lower bounds and upper bounds is usually referred to as the *circulation problem*, and is polynomially solvable [18].

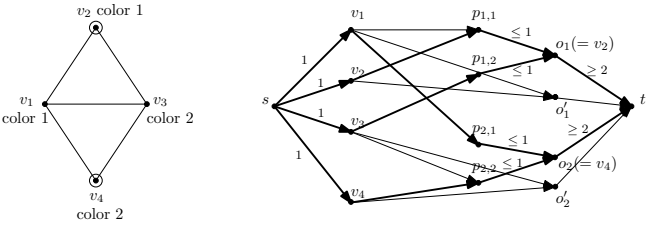


Fig. 2. The flow network construction. On the left is the original graph, $I = \{v_2, v_4\}$, $\ell = 2$. On the right is the corresponding flow network. Thick edges denote a feasible flow of value $|I|\ell = 4$.

Once G_i and I pass F-TEST, we try to redistribute those vertices that cause color conflicts. We do so by a bipartite matching test M-TEST(G_i, I) which returns two vertex sets S and S' that are useful later. Concretely, we test whether there exists a matching

in the bipartite graph $B(I - C, C - I; E_{G_i}(I - C; C - I))$ for each color class C such that all vertices in $C - I$ are matched. If such matchings can be found for all the colors, we say that the M-TEST is passed. Note that all these matchings together give a spanning star forest such that each star is polychromatic. However, this does not guarantee that the cardinality constraint is preserved. The crucial fact here is that I passes *both* F-TEST and M-TEST. In Lemma 2 we formally prove that there exists a valid spanning star forest with nodes in I as star centers if and only if G_i and I pass both F-TEST and M-TEST. To actually find a valid spanning star forest, we can again use the network flow construction in F-TEST but without the o'_j nodes. If M-TEST fails, we know that for some color class C, there exists a subset $S \subseteq C - I$ such that the size of its neighbor set $|N_B(S)|$ is less than $|S|$ by Hall's theorem [18]. In this case, M-TEST returns $(S, N_B(S))$; such a set S can be found by a maximum matching algorithm. Then we update the independent set $I \leftarrow I - N_B(S) + S$; we show that I is still an independent set in Lemma 1. Finally, we add nodes to I arbitrarily until it becomes a maximal independent set. Then, we start the next iteration with the new I. Since $|S| > |N_B(S)|$, we increase $|I|$ by at least one in each iteration. So the algorithm terminates in $\leq n$ iterations.

Before proving that Algorithm 1 is guaranteed to succeed when $w(e_i) = d^*$, we prove the two lemmas left in the description of our algorithm. The first lemma ensures that I is always an independent set.

Lemma 1. *The new set $I \leftarrow I - S' + S$ obtained in each update is still an independent set in G_i.*

Proof. Since all vertices in S have the same color, there is no edge among them. Therefore, we only need to prove that there is no edge between $I - S'$ and S, which is trivial since $S' = N_B(S)$. □

The second lemma guarantees that we find a feasible solution if both tests are passed.

Lemma 2. *Given $G_i = G(V, E_i)$ and I, a maximal independent set of G_i, both F-TEST(G_i,I) and M-TEST(G_i,I) are passed if and only if there exists a valid spanning star forest in G_i with nodes in I being star centers.*

Proof. The "if" part is trivial. We only prove the "only if" part. Suppose G_i and I pass both F-TEST(G_i,I) and M-TEST(G_i,I). Consider a semi-valid spanning star forest obtained in G_i after F-TEST. We delete a minimal set of leaves to make it a valid star (not necessarily spanning) forest F. Consider the bipartite graph $B(I - C; C - I, E_{G_i}(I - C; C - I))$ for each color class C. We can see $F \cap B$ is a matching in B. Since I passes M-TEST, we know that there exists a maximum matching such that all nodes in $C - I$ can be matched. If we use the Hungarian algorithm to compute a maximum matching with $F \cap B$ as the initial matching, the nodes in $I - C$ which are originally matched will still be matched in the maximum matching due to the property of the alternating path augmentation[1]. Therefore, the following invariants are maintained: each

[1] Recall that an augmenting path P (with respect to matching M) is a path starting from unmatched node, alternating between unmatched and matched edges and ending also at an unmatched node (for example, see [23]). By taking the symmetric difference of P and M, which we call augmenting on P, we can obtain a new matching with one more edge.

star is polychromatic and has at least ℓ colors. By applying the above maximum matching computation for each color class, we obtain a valid spanning star forest in G_i. □

Finally, we prove that Algorithm 1 is guaranteed to succeed on G_{i*} for the maximal index i^* such that $w(e_{i^*}) = d^*$, where d^* is the optimal cluster diameter of any valid spanning star forest of G.

Lemma 3. *Algorithm 1 will succeed on G_{i^*}.*

Proof. Suppose $\mathcal{C}^* = \{C_1^*, \ldots, C_{k^*}^*\}$ is the set of clusters in the optimal clustering with cluster diameter d^*. Since G_{i*} include all edges of weights no more than d^*, each C_j^* induces a clique in G_{i^*} for all $1 \leq j \leq k^*$, thus it contains at most one node in any independent set. Therefore, any maximal independent set I in G_{i^*} can pass F-TEST(G_{i^*}, I), and we only need to argue that I will also pass M-TEST(G_{i^*}, I). Each update to the independent set I increases the size of I by at least 1 and the maximum size of I is k^*. When $|I| = k^*$, each C_j^* contains exactly one node in I and this I must be able to pass M-TEST(G_{i^*}, I). So Algorithm 1 must succeed in some iteration. □

By Lemma 3, the cost of the spanning star forest found by Algorithm 1 is at most d^*. Since the cost of the optimal spanning forest is at least $d^*/2$, we obtain a 2-approximation.

Theorem 1. *There is a polynomial-time 2-approximation for ℓ-DIVERSITY.*

3 The Lower Bound

We show that ℓ-DIVERSITY is NP-hard to approximate within a factor less than 2 even when there are only 3 colors. Note that if there are 2 colors, the problem can be solved in polynomial time by computing perfect matchings in the threshold graphs.

Theorem 2. *There is no polynomial-time approximation algorithm for ℓ-DIVERSITY that achieves an approximation factor less than 2 unless $P = NP$.*

4 Dealing with Unqualified Inputs

For the ℓ-DIVERSITY problem, a feasible solution may not exist depending on the input color distribution. The following simple lemma gives a necessary and sufficient condition for the existence of a feasible solution.

Lemma 4. *There exists a feasible solution for ℓ-DIVERSITY if and only if the number of nodes with the same color c is at most $\lfloor \frac{n}{\ell} \rfloor$ for each color c.*

To cluster an instance without a feasible solution, we must exclude some nodes as *outliers*. The following lemma characterizes the minimum number of outliers.

Lemma 5. *Let $C_1, C_2, \ldots, C_k$ be the color classes sorted in the non-increasing order of their sizes.*

1. *Let p be the maximum integer satisfying $\sum_{i=1}^{k} \min(p, |C_i|) \geq p\ell$. The minimum number of outliers is given by $q = \sum_{i=1}^{k} \max(0, |C_i| - p)$ and p is the number of clusters when we exclude q outliers.*
2. *$p\ell \leq n - q < p(\ell + 1)$.*

With lemma 5 at hand, it is natural to consider the following optimization problem: find an ℓ-DIVERSITY solution by clustering $n - q$ points such that the maximum cluster radius is minimized. We call this problem ℓ-DIVERSITY-OUTLIERS. From Lemma 5 we can see that p is independent on the metric and can be computed in advance. In addition, implicit from Lemma 5 is that the number of outliers of each color is also fixed, but we need to decide *which* points should be chosen as outliers.

In the fortunate case where we have a color class C with exactly p nodes, we know that there is exactly one node of C in each cluster of any feasible solution. By using a similar flow network construction used in F-TEST, we can easily get a 2-approximation using C as the cluster centers. However, the problem becomes much more difficult when the sizes of all color classes are different from p.

4.1 A Constant Approximation

We first define some notations. We call color classes of size larger than p *popular colors* and nodes having such colors *popular* nodes. Other color classes have at most p nodes, and these nodes are *unpopular*. We denote the set of popular nodes by $\mathcal{P}$ and the set of unpopular nodes by $\mathcal{N}$. Note that after removing the outliers, each popular color has exactly p nodes and each cluster will contain the same number of popular nodes. Let z be the number of popular nodes each cluster contains. We denote by G^d the power graph of G in which two vertices u, v are adjacent if there is path connecting u and v with at most d edges. The length of the edge (u, v) in G^d is set to be $\mathsf{dist}_G(u, v)$. Before describing the algorithm, we need the following simple lemma.

Lemma 6. *For any connected graph G, G^3 contains a Hamiltonian cycle which can be found in linear time.*

The algorithm still adopts the thresholding method, that is, we add edges one by one to get graphs $G_i = (V, E_i = \{e_1, e_2, \ldots, e_i\})$, for $i = 1, 2, \ldots$, and in each G_i, we try to find a valid star forest that spans G_i except q outliers. Let d^* be the diameter of the optimal solution that clusters $n - q$ points, and i^* be the maximum index such that $w(e_i) = d^*$. Let $G_i[\mathcal{N}]$ be the subgraph of G_i induced by all unpopular nodes. We define the ball of radius r around v to be $\mathsf{B}(v, r) = \{u \mid u \in \mathcal{N} \wedge \mathsf{dist}_G(v, u) \leq r\}$. For each G_i, we run the Algorithm: ℓ-DIVERSITY-OUTLIERS(G_i) (see below). We proceed to G_{i+1} when the algorithm claims failure.

The high level idea of the algorithm is as follows: Our goal is to show that the algorithm can find a valid star forest spanning $n - q$ nodes in $G_{i^*}^{28}$. It is not hard to see that this gives us an approximation algorithm with factor $28 \times 2 = 56$. First, we notice that F-TEST can be easily modified to work for the outlier version by excluding all o'_j nodes and testing whether there is a flow of value $n - q$. However, the network flow construction needs to know in advance the set of candidates of cluster centers. For this purpose, we attempt to attach p new nodes which we call *virtual centers* to G_i which serve as the candidates of cluster centers in F-TEST. In the ideal case, if these virtual centers can be distributed such that each of them is attached to a distinct optimal cluster, F-TEST can easily produce a 2-approximation. Since the optimal clustering is not known, this is very difficult in general. However, we show there is way to carefully distribute the virtual centers such that there is a perfect matching between these virtual centers and the

optimal cluster centers and the longest matching edge is at most $27d^*$, which implies that our algorithm can find a valid spanning star forest in $G_{i^*}^{27+1} = G_{i^*}^{28}$

Algorithm: ℓ-*DIVERSITY-OUTLIERS*(G_i).

1. If $G_i[\mathcal{N}]$ has a connected component with $< \ell - z$ nodes, we declare failure.
2. Pick an arbitrary unpopular node v such that $|\mathsf{B}(v, w(e_i))| \geq \ell - z$ and delete all vertices in this ball; repeat until no such node exists. Then, pick an arbitrary unpopular node v and delete all vertices in $\mathsf{B}(v, w(e_i))$; repeat until no unpopular node is left. Let $\mathsf{B}_1, \mathsf{B}_2, \ldots, \mathsf{B}_k$ be the balls created during the process. If a ball contains at least $\ell - z$ unpopular nodes, we call it *big*. Otherwise, we call it *small*.
3. In $G_i[\mathcal{N}]$, shrink each B_j into a single node b_j. A node b_j is *big* if B_j is big and *small* otherwise. We define the *weight* of b_j to be $\mu(b_j) = \frac{|\mathsf{B}_j|}{\ell - z}$. Let the resulting graph with vertex set $\{b_j\}_{j=1}^k$ be D_i.
4. For each connected component C of D_i, do
   (a) Find a spanning tree T_C of C^3 such that all small nodes are leaves. If this is not possible, we declare failure.
   (b) Find (by Lemma 6) a Hamiltonian cycle $P = \{b_1, b_2, \ldots, b_h, b_{h+1} = b_1\}$ over all non-leaf nodes of C such that $\mathsf{dist}_{D_i}(b_j, b_{j+1}) \leq 9w(e_i)$.
5. We create a new color class U of p nodes which will serve as "virtual centers" of the p clusters. These virtual centers are placed in G_i "evenly" as follows. Consider each connected component C in D_i and the corresponding spanning tree T_C of C^3. For each non-leaf node b_j in T_C, let $L(b_j)$ be the set of leaves connected to b_j in T_C, and let $\eta(b_j) = \mu(b_j) + \sum_{b_x \in L(b_j)} \mu(b_x)$ and $\delta_j = \sum_{x=1}^{j} \eta(b_x)$. We attach $\lfloor \delta_i \rfloor - \lfloor \delta_{i-1} \rfloor$ virtual centers to the center of B_i by zero weight edges. If the total number of virtual centers used is not equal to p, we declare failure. Let H_i be the resulting graph (including all popular nodes, unpopular nodes and virtual centers).
6. Find a valid star forest in H_i^{28} using U as centers, which spans $n - q$ nodes (not including the nodes in U) by using F-TEST. If succeeds, we return the star forest found, otherwise we declare failure.

4.2 Analysis of the Algorithm

We show that the algorithm succeeds on G_{i^*}. Since we perform F-TEST on $H_{i^*}^{28}$ in which each edge is of length $\leq 28d^*$, the radius of each cluster is at most $28d^*$. Therefore, the approximation ratio is 56.

Let H_{i^*} be the graph obtained by adding virtual centers to G_{i^*} as described above. Let $\mathcal{C}^* = \{C_1^*, \ldots, C_p^*\}$ be the optimal clustering. Let $I^* = \{\nu_1^*, \ldots, \nu_p^*\}$ be the set of cluster centers of $\mathcal{C}^*$ where ν_i^* is the center of C_i^*. We denote the balls grown in step 2 by $\mathsf{B}_1, \ldots, \mathsf{B}_k$. Let ν_i be the center of B_i.

The algorithm may possibly fail in step 1, step 4(a), step 5 and step 6. Obviously G_{i^*} can pass step 1. Therefore, we only check the other three cases.

Step 4(a) : We prove that the subgraph induced by all big nodes are connected in C^3. Indeed, we claim that each small node is adjacent to at least one big node in C from

which the proof follows easily. Now we prove the claim. Suppose b_j is a small node and all its neighbors are small. We know that in $G_{i^*}[\mathcal{N}]$, ν_j has at least $\ell - z - 1$ neighbors because ν_j is an unpopular node and thus belongs to some optimal cluster. So we could form a big ball around ν_j, thus contradicting to the fact that ν_j is in a small ball. To find a spanning tree with all small nodes as leaves, we first assign each small node to one of its adjacent node arbitrarily and then compute a tree spanning all the big nodes.

Step 5 : We can see that in each connected component C (with big nodes $b_1, \ldots, b_h$) in D_{i^*}, the total number of virtual centers we have placed is $\sum_{i=1}^{h}(\lfloor \delta_i \rfloor - \lfloor \delta_{i-1} \rfloor) = \lfloor \delta_h \rfloor = \lfloor \sum_{x=1}^{h} \eta(b_x) \rfloor = \lfloor \sum_{b_j \in C} \mu(b_j) \rfloor = \left\lfloor \frac{|C|}{l-z} \right\rfloor$ where $|C| = \sum_{b_j \in C} |\mathsf{B}_j|$, the number of nodes in the connected component of $G_{i^*}[\mathcal{N}]$ corresponding to C. This is at least the number of clusters created for the component C in the optimal solution. Therefore, we can see the total number of virtual centers created is at least p. On the other hand, from Lemma 5(2), we can see that $p(\ell - z) \leq |\mathcal{N}| < (p+1)(\ell - z)$. Hence, $p = \left\lfloor \frac{|\mathcal{N}|}{\ell - z} \right\rfloor = \left\lfloor \frac{\sum_C |C|}{\ell - z} \right\rfloor \geq \sum_{\mathrm{C}} \left\lfloor \frac{|C|}{\ell - z} \right\rfloor$. where the summation is over all connect components. So, we prove that exactly p virtual centers were placed in G_{i^*}.

Step 6 : We only need to show that there is a perfect matching M between U and the set of optimal centers I^* in $H_{i^*}^{27}$. We consider the bipartite subgraph $Q(U, I^*, E_{H_{i^*}^{27}}(U, I^*))$. From Hall's theorem, it suffices to show that $|N_Q(S)| \geq |S|$ for any $S \subseteq U$, which can be implied by the following lemma.

Lemma 7. *For any $S \subseteq U$, the union of the balls of radius $27d^*$ around the nodes of S, i.e, $\bigcup_{u \in S} \mathsf{B}(u, 27d^*)$, intersects at least $|S|$ optimal clusters in $\mathcal{C}^*$.*

Theorem 3. *There is a 56-approximation for ℓ-DIVERSITY-OUTLIERS.*

5 Further Directions

This work results in several open questions. First, as in [1], we could also try to minimize the sum of the radii of the clusters. However, this seems to be much more difficult, and we leave it as an interesting open problem. Another open problem is to design constant approximations for the problem with any fixed number of outliers, that is, for any given number k, find an optimal clustering if at most k outliers can be removed.

As mentioned in the introduction, our work can be seen as a stab at the more general problem of clustering under instance-level hard constraints. Although arbitrary CL (cannot-link) constraints seems hard to approximate with respect to minimizing the number of clusters due to the hardness of graph coloring [10], other objectives and special classes of constraints, e.g. diversity constraints, may still admit good approximations. Besides the basic ML and CL constraints, we could consider more complex constraints like the rules proposed in the Dedupalog project [4]. One example of such rules says that whenever we cluster two points a and b together, we must also cluster c and d. Much less is known for incorporating these types of constraints into traditional clustering problems and we expect it to be an interesting and rich further direction.

References

1. Aggarwal, G., Feder, T., Kenthapadi, K., Khuller, S., Panigrahy, R., Thomas, D., Zhu, A.: Achieving anonymity via clustering. In: PODS, pp. 153–162 (2006)
2. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Anonymizing tables. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 246–258. Springer, Heidelberg (2004)
3. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: Ranking and clustering. J. ACM 55(5), 1–27 (2008)
4. Arasu, A., Ré, C., Suciu, D.: Large-scale deduplication with constraints using Dedupalog. In: ICDE, pp. 952–963 (2009)
5. Bairoch, A., Apweiler, R.: The SWISS-PROT protein sequence data bank and its supplement TrEMBL. Nucleic acids research 25(1), 31 (1997)
6. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning 56(1), 89–113 (2004)
7. Beresford, A., Stajano, F.: Location privacy in pervasive computing. IEEE Pervasive Computing, 46–55 (2003)
8. Wong, R.C.-W., Li, J., Fu, A.-C., Wang, K.: (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In: SIGKDD, pp. 754–759 (2006)
9. Charikar, M., Khuller, S., Mount, D., Narasimhan, G.: Algorithms for facility location problems with outliers. In: SODA, pp. 642–651 (2001)
10. Davidson, I., Ravi, S.: Intractability and clustering with constraints. In: ICML, pp. 201–208 (2007)
11. Dwork, C., Naor, M., Reingold, O., Rothblum, G., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: STOC, pp. 381–390 (2009)
12. Feldman, D., Fiat, A., Kaplan, H., Nissim, K.: Private coresets. In: STOC, pp. 361–370 (2009)
13. Ghinita, G., Karras, P., Kalnis, P., Mamoulis, N.: Fast data anonymization with low information loss. In: VLDB, pp. 758–769 (2007)
14. Giotis, I., Guruswami, V.: Correlation clustering with a fixed number of clusters. In: SODA, pp. 1176–1185 (2006)
15. Hoppner, F., Klawonn, F., Platz, R., Str, S.: Clustering with Size Constraints. Computational Intelligence Paradigms: Innovative Applications (2008)
16. Ji, X.: Graph Partition Problems with Minimum Size Constraints. PhD thesis, Rensselaer Polytechnic Institute (2004)
17. Kifer, D., Gehrke, J.: Injecting utility into anonymized datasets. In: SIGMOD, pp. 217–228 (2006)
18. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms, 4th edn. Springer, Heidelberg (2007)
19. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: ICDE, p. 25 (2006)
20. Li, J., Yi, K., Zhang, Q.: Clustering with diversity (2010), http://arxiv.org/abs/1004.2968
21. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: ICDE, p. 24 (2006)
22. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: PODS, pp. 223–228 (2004)
23. Alsuwaiyel, M.H.: Algorithms: Design Techniques and Analysis. World Scientific, Singapore (1998)

24. Park, H., Shim, K.: Approximate algorithms for k-anonymity. In: SIGMOD (2007)
25. Samarati, P.: Protecting respondents' identities in microdata release. TKDE 13(6), 1010–1027 (2001)
26. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: ICML, pp. 1103–1110 (2000)
27. Wagstaff, K., Cardie, C., Schroedl, S.: Constrained k-means clustering with background knowledge. In: ICML, pp. 577–584 (2001)
28. Xiao, X., Tao, Y.: Anatomy: Simple and effective privacy preservation. In: VLDB, pp. 139–150 (2006)
29. Xiao, X., Tao, Y.: m-invariance: Towards privacy preserving re-publication of dynamic datasets. In: SIGMOD, pp. 689–700 (2007)
30. Xiao, X., Yi, K., Tao, Y.: The hardness and approximation algorithms for l-diversity. In: EDBT (2010)
31. Xing, E., Ng, A., Jordan, M., Russell, S.: Distance metric learning, with application to clustering with side-information. In: NIPS, pp. 505–512 (2003)

New Data Structures for Subgraph Connectivity

Ran Duan*

University of Michigan, Ann Arbor
duanran@umich.edu

Abstract. We study the "subgraph connectivity" problem for undirected graphs with sublinear vertex update time. In this problem, we can make vertices active or inactive in a graph G, and answer the connectivity between two vertices in the subgraph of G induced by the active vertices. Two open problems in subgraph connectivity are solved in this paper. We give the first subgraph connectivity structure with worst-case sublinear time bounds for both updates and queries. Our worst-case subgraph connectivity structure supports $\tilde{O}(m^{4/5})$ update time, $\tilde{O}(m^{1/5})$ query time and occupies $\tilde{O}(m)$ space, where m is the number of edges in the whole graph G.

In the second part of our paper, we describe another dynamic subgraph connectivity structure with amortized $\tilde{O}(m^{2/3})$ update time, $\tilde{O}(m^{1/3})$ query time and linear space, which improves the structure introduced by [Chan, Pătraşcu, Roditty, FOCS'08] that takes $\tilde{O}(m^{4/3})$ space.

1 Introduction

In this paper, we study the fully dynamic connectivity problem with vertex updates on graphs, which is also known as "subgraph connectivity". More precisely, given an undirected graph $G = (V, E)$, we can switch the status of every vertex to be active or inactive, and the structure can answer the connectivity between any two active vertices on the subgraph of G induced by the active vertices. We define $n = |V|$ and $m = |E|$. The dynamic subgraph model was introduced by Frigioni and Italiano [11], in which a vertex and all edges associated with it can be made active or inactive in one update without adding or removing those edges one by one. They gave a dynamic subgraph connectivity structure having poly-logarithmic vertex update time in planar graphs. Recently, Chan, Pătraşcu and Roditty [2] gave a subgraph connectivity structure for general graphs supporting $\tilde{O}(m^{2/3})$[1] vertex update time with $\tilde{O}(m^{4/3})$ space, which improves the result given by Chan [1] having $\tilde{O}(m^{0.94})$ update time and linear space. However, the update time bounds for both of these structures are amortized. In this paper, we present the first structure having worst-case $o(m)$ update and query time on dynamic subgraph connectivity, and also improve the amortized structure in [2] to linear space with the same update and query time bounds.

* This work is supported by NSF CAREER grant no. CCF-0746673 and a grant from the US-Israel Binational Science Foundation.

[1] In this paper, $\tilde{O}(\cdot)$ hides poly-logarithmic factors.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 201–212, 2010.

Dynamic connectivity with edge updates is the most basic problem among this kind of dynamic structures and is well studied. Thorup has introduced a linear space structure supporting $O(\log^2 n)$ amortized update time [22]. With this structure, we can get a trivial dynamic subgraph connectivity structure with amortized vertex update time $\tilde{O}(n)$. Then two hard directions related to this problem arise: dynamic subgraph connectivity with sublinear vertex update time, and dynamic structures with worst-case edge/vertex update time bounds.

As mentioned above, for the subgraph connectivity in general graphs, Chan [1] has given a linear space structure with $\tilde{O}(m^{4\omega/(3\omega+3)} \approx m^{0.94})$ update time and $\tilde{O}(m^{1/3})$ query time. Here $\omega \approx 2.376$ is the exponent for the matrix multiplication time bound [3]. Chan, Pătraşcu and Roditty [2] have improved the update time to $\tilde{O}(m^{2/3})$ and gotten rid of the fast matrix multiplication operations.

However, the dynamic structures mentioned above all have amortized update time. In general, worst-case dynamic structures have much worse time bounds than amortized structures. Frederickson has given a dynamic minimum spanning tree structure which supports edge updates in $O(m^{1/2})$ time in the worst-case scenario [10]. Eppstein, Galil, Italiano, and Nissenzweig [9] have improved the time bound to $O(n^{1/2})$ by a technique called "sparsification". Improving this time bound is still a major challenge in dynamic graph algorithms [16]. Using those worst-case edge update structures, we give a natural generalization in this paper– the first dynamic subgraph connectivity structure with sublinear vertex update time in the worst-case scenario.

Our Results. In the first part of this paper, we will describe a worst-case dynamic subgraph connectivity structure with $\tilde{O}(m^{4/5})$ vertex update time and $\tilde{O}(m^{1/5})$ query time. We will utilize the worst-case edge update structure [9] as a component and maintain a multi-level hierarchy instead of the two-level one in [2]. In general, we will get faster update time for this structure if there are faster worst-case edge update connectivity structures. In the second part of this paper, we will describe a new linear space subgraph connectivity structure with $\tilde{O}(m^{2/3})$ amortized vertex update time and $\tilde{O}(m^{1/3})$ query time.

Techniques. The best worst-case edge update connectivity structure [9] so far has $O(n^{1/2})$ update time, much larger than the polylogarithmic amortized edge update structure [13,22]. Inspiring by [2], we will divide vertices into several levels by their degrees. In "lower" levels having small degree bounds, we maintain an edge update connectivity structure for the subgraph on active vertices at these levels. In "higher" levels having large degree bounds and small numbers of vertices, we only keep the subgraph at those levels and run a BFS to obtain all the connected components after an update. To reflect the connectivity between high-level vertices through low-level vertices, we will add two types of artificial edges to the high-level vertices. (a). In the "path graph", update on every vertex will change the edge set, but the number of edges changed is only linear to the degree of that vertex. (b). In the "complete graph", only low-level vertex updates will change the edge set, but the number of edges changed is not linear to the

degree. In our structure, we only use the "complete graph" between top levels and bottom levels to bound the update time.

Related Results. Data structures on dynamic graphs are widely applicable in many areas. A huge amount of papers in algorithms and theory focused on many basic properties on dynamic graphs, such as connectivity [9,11,1,2,13,21,22,10], all-pair distances [4,18,23,14], minimum spanning trees [12,13,10], and reachability in directed graphs [17,20,19,6,15]. Also there are researches on connectivity and shortest path problem on failure prone graphs, in which queries are given with a constant number of vertex/edge failures. [5,7,16,8]

2 Basic Structures

In this section, we will define several dynamic structures as elements of the main structures. If we want to keep the connectivity of some vertex set V_1 through a disjoint set V_0, some "artificial edges" may need to be added into V_1. For every spanning tree in V_0, the vertices in V_1 adjacent to this spanning tree need to be connected. We will use the ET-tree ideas from Henzinger and King's paper [12] to make such artificial edges efficiently dynamic when the spanning forest of V_0 changes. Here the artificial edges of V_1 associated with a spanning tree in V_0 will form a path ordered by the Euler Tour of that tree.

2.1 Euler Tour List

For a tree T, let $L(T)$ be a list of its vertices encountered during an Euler tour of T [12], where we only keep *any one* of the occurrences of each vertex. Note that $L(T)$ can start at any vertex in T. Now we count the number of cut/link operations on the Euler tour lists when we cut/link trees. One may easily verify the following theorem:

Theorem 1. *When we delete an edge from T, T will be split into two subtrees T_1 and T_2. We need at most 2 "cut" operations to split $L(T)$ into 3 parts, and at most 1 "link" operation to form $L(T_1)$ and $L(T_2)$.*

When we add an edge to link two tree T_1 and T_2 into one tree T, then we need to change the start or end vertices of $L(T_1)$ and $L(T_2)$ and link them together to get $L(T)$, which will take at most 5 "cut/link" operations.

2.2 Adjacency Graph

In a graph $G = (V, E)$, let $V_0, V_1, V_2, ..., V_k$ be disjoint subsets of V, and let F be a forest spanning the connected components of the subgraph of G induced by the active vertices of V_0. We will construct a structure $R(G, F, V_1, V_2, ..., V_k)$ containing artificial edges on the active vertices of the sets $V_1, V_2, ..., V_k$ which can represent the connectivity of these vertices through V_0.

Definition 1. *For $1 \leq i \leq k$, the* active adjacency list $A_G(v, V_i)$ *of a vertex $v \in V_0$ is the list of active vertices in V_i which are adjacent to v in G. The* active adjacency list $A_G(T, V_i)$ *induced by a tree $T \in F$ is the concatenation of*

the lists $A_G(v_1, V_i), A_G(v_2, V_i), ..., A_G(v_k, V_i)$ *where* $L(T) = (v_1, v_2, ..., v_k)$. *Note that a vertex of* V_i *can appear multiple times in* $A_G(T, V_i)$.

Definition 2. *Given a list* $l = (v_1, v_2, ..., v_k)$ *of vertices, define the edge set* $P(l) = \{(v_i, v_{i+1}) | 1 \leq i < k\}$.

Definition 3. *In the structure* $R(G, F, V_1, V_2, ..., V_k)$, *for a tree* $T \in F$, *we maintain the list* $A_G(T)$ *of active vertices which is the concatenation of the lists* $A_G(T, V_1), A_G(T, V_2), ..., A_G(T, V_k)$. *Then the set of artificial edges in* $R(G, F, V_1, V_2, ..., V_k)$ *is the union* $\bigcup_{T \in F} P(A_G(T))$. *We call the edges connecting different* $A_G(T, V_i)$ $(1 \leq i \leq k)$ *"inter-level edges". So the degree of a vertex* v *of* V_i $(1 \leq i \leq k)$ *in* $R(G, F, V_1, V_2, ..., V_k)$ *is at most twice its degree in* G, *and the space of this structure is linear to* G.

We can see that deleting a vertex in l will result in deleting at most two edges and inserting at most one edge in $P(l)$, and inserting a vertex in l will result in inserting at most two edges and deleting at most one edge in $P(l)$. Also, one can easily verify the following properties of the adjacency graph:

Note 1. For a spanning tree $T \in F$, the vertices in $A_G(T, V_i)$ are connected by the subset of $R(G, F, V_1, V_2, ..., V_k)$ induced only by V_i, for all $1 \leq i \leq k$.

Lemma 1. *For any two active vertices* u, v *in* $V_1 \cup V_2 \cup ... \cup V_k$, *if there is a path with more than one edge connecting them, whose intermediate vertices are active and in* V_0, *then they are connected by the edges* $R(G, F, V_1, V_2, ..., V_k)$.

Also if u, v *are connected in* $R(G, F, V_1, V_2, ..., V_k)$, *they are connected in the subgraph of* G *induced by the active vertices.*

Lemma 2. *The cost needed to maintain this structure:*

1. *Making a vertex* v *active or inactive in* $V_1 \cup V_2 \cup ... \cup V_k$ *will require inserting or deleting at most* $O(min(deg_G(v), |V_0|))$ *edges in* $R(G, F, V_1, V_2, ..., V_k)$. *(Here* $deg_G(v)$ *denotes the degree of* v *in the graph* G.*)*
2. *Adding or removing an edge in* F *will require inserting or deleting* $O(k)$ *edges to this structure.*
3. *Making a vertex* $v \in V_0$ *active or inactive will require inserting or deleting* $O(k \cdot deg_G(v))$ *edges.*
4. *Inserting or deleting an "inter-level" edge* (u, v) *in* G *where* $u \in V_0$, $v \in V_1 \cup V_2 \cup ... \cup V_k$ *will require inserting or deleting at most 3 edges in* $R(G, F, V_1, V_2, ..., V_k)$. *(*$G$ *may be not the original graph, but another dynamic graph.)*

2.3 ET-List for Adjacency

Here we describe another data structure for handling adjacency queries among a dynamic spanning tree F and a disjoint vertex set V_1. By this structure, when we intend to obtain all the vertices in V_1 adjacent to a tree $T \in F$, we do not need to check all the edges connecting T to V_1, but only check whether v is adjacent to T for all $v \in V_1$. This takes $O(|V_1|)$ time for finding all such vertices. Note that since this structure keeps all the vertices in V_1 no matter whether they are active or not, so we do not need to update it when switching a vertex in V_1.

Theorem 2. *Let $G = (V, E)$ be a graph and V_0, V_1 be two disjoint subsets of V. Let F be a spanning forest on the subgraph of G induced by the active vertices of V_0. There is a data structure $ET(G, F, V_1)$ with linear size that accepts edge inserting/deleting updates in F. Given a vertex $v \in V_1$ (active or inactive) and a tree $T \in F$, we can answer whether they are adjacent in G in constant time. The update time for a vertex v in V_0 of this structure is $O(deg_G(v)|V_1|)$.*

Proof. In $ET(G, F, V_1)$, for every vertex $v \in V_1$ and every $T \in F$, we keep a list of vertices in T adjacent to v ordered by $L(T)$. From Theorem 1, when we link two trees or cut a tree into two subtrees in F, it takes $O(V_1)$ time to merge or split the lists for all $v \in V_1$. When a vertex in V_0 is turned active or inactive, we need to add/delete $deg_G(v)$ edges in F and add/delete that vertex in the lists for all $v \in V_1$. The space will be $O(m)$ since every edge will contribute at most one appearance of vertex in the lists.

3 Dynamic Subgraph Connectivity with Sublinear Worst-Case Update Time

In this section, we will describe our worst-case dynamic subgraph connectivity structure with sublinear update time. We divide the vertices into several levels by their degrees. The structure of adjacency graph in Section 2.2 will be used to reflect the connectivity between high-level vertices through low-level vertices. We will use the dynamic spanning tree structure of $O(n^{1/2})$ worst-case edge update time [9] to keep the connectivity of vertices in low-levels of lower degree bounds. However, in high-levels with high degree bounds, we only store the active vertices and edges and run a BFS after each update to obtain the new spanning trees.

Theorem 3. *Given a graph $G = (V, E)$, there exists a dynamic subgraph connectivity structure occupying $\tilde{O}(m)$ space and taking $\tilde{O}(m^{6/5})$ preprocessing time. We can switch every vertex to be "active" or "inactive" in this structure in $\tilde{O}(m^{4/5})$ time, and answer the connectivity between any pair of vertices in the subgraph of G induced by the active vertices in $\tilde{O}(m^{1/5})$ query time.*

3.1 The Structure

First we divide all the vertices of G into several parts based on their degrees in the whole graph G, so the sets are static.

- V_A: The set of vertices of degrees less than $m^{1/5}$
- V_B: The set of vertices v satisfying $m^{1/5} \leq deg_G(v) < m^{3/5}$.
- V_C: The set of vertices v satisfying $m^{3/5} \leq deg_G(v) < m^{4/5}$.
- V_D: The set of vertices v satisfying $deg_G(v) \geq m^{4/5}$.

So we can see that $|V_B| \leq 2m^{4/5}, |V_C| \leq 2m^{2/5}, |V_D| \leq 2m^{1/5}$.

In order to get more efficient update time, we continue to partition the set V_B into $V_0, V_1, V_2, ..., V_k$ where $k = \lfloor \frac{2}{5} \log_2 m \rfloor$ and:

$$V_i = \{v | v \in V_B, 2^i m^{1/5} \leq deg_G(v) < 2^{i+1} m^{1/5}\}, \forall 0 \leq i \leq k \tag{1}$$

Thus, $|V_i| \leq 2^{1-i}m^{4/5}$. For all the disjoint vertex sets $V_A, V_0, V_1, ..., V_k, V_C, V_D$ ordered by their degree bounds, we say that a vertex u is *higher than* a vertex v if u is in the set of higher degree bound than v.

For the set V_A, the following structure will be built to keep the connectivity between vertices in other sets through vertices of V_A:

- Maintain a dynamic spanning forest F_A on the subgraph of G induced by the active vertices of V_A, which will support $O(\sqrt{n})$ edge update time. [9]
- Maintain the edge set (and the structure) $E_A = R(G, F_A, V_0, V_1, ..., V_k, V_C)$.
- Maintain the structures $ET(G, F_A, V_C)$,$ET(G, F_A, V_D)$ so that we can find the vertices of V_C and V_D (including active and inactive) adjacent to a tree T of F_A in G in $O(|V_C|)$ time by Theorem 2. Denote the vertices of V_C and V_D adjacent to T by $V_C(T)$ and $V_D(T)$, respectively.
- For every spanning tree $T \in F_A$, arbitrarily choose an **active** vertex $u_T \in V_B$ which is adjacent to T in G (if there is one). Call it the "representative" vertex of T. Define the edge set $\bar{E}_T = \{(u, v)|u \in V_C(T) \cup V_D(T), v \in V_D(T)\} \cup \{(u_T, v)|v \in V_D(T)\}$.
- Define $G_0 = (V, E \cup E_A \cup \bigcup_{T \in F_A} \bar{E}_T)$. Note that E_A only contains edges connecting active vertices, but $\bar{E}_T$ may contain edges associate with inactive vertices. When considering the connectivity of G_0, we only consider the subgraph of G_0 induced by the active vertices and ignore the inactive vertices.

We have added artificial edges on the vertices of V_B, V_C, V_D to G_0 so that the subgraph of G_0 induced by the active vertices of these sets can represent the connectivity in the dynamic graph G. Note that we do not store the set $\bar{E}_T$ for every $T \in F_A$, but only store the final graph G_0 to save space. We can get every $\bar{E}_T$ efficiently from the adjacency lists.

Then we will build structures for the connectivity on $V_B \cup V_C \cup V_D$ through $V_0, ..., V_k$. For $i = 0$ to k, perform the following two steps:

1. Maintain a dynamic spanning forest F_i on the subgraph of G_i induced by the active vertices of V_i. The structure will support $O(\sqrt{|V_i|}) = O(m^{2/5}/2^{i/2})$ edge update time. [9]
2. Maintain the edge set $E_{i+1} = R(G_i, F_i, V_{i+1}, ..., V_k, V_C, V_D)$, and define the graph $G_{i+1} = (V, E(G_i) \cup E_{i+1})$, where $E(G_i)$ is the set of edges in G_i.

We denote $H = G_{k+1}$ which contains all the artificial edge. Note that only the edges connecting vertices higher than V_i will be added to G_{i+1}, so the spanning forest F_i (F_A) still spans the connected components of the subgraph of H induced by the active vertices of V_i (V_A), and also $E_A = R(H, F_A, V_0, V_1, ..., V_k, V_C)$, $E_{i+1} = R(H, F_i, V_{i+1}, ..., V_k, V_C, V_D)$ for all $0 \leq i \leq k$.

Discussion: Why we need $\bar{E}_T$ but not simply construct $E_A = R(G, F_A, V_0, ..., V_k, V_C, V_D)$? Since there are no specific bounds for $|V_D|$ and the number of spanning trees in F_A, if $E_A = R(G, F_A, V_0, ..., V_k, V_C, V_D)$, from Lemma 2(1), the update time may become linear when we switch a vertex in V_D. Remind that $\bar{E}_T$ contains the edges connecting active and inactive vertices in V_D, so we do not need to change the edge sets $\bar{E}_T$ when switching a vertex of V_D.

When we consider the connectivity of vertices of V_C and V_D in H after an update, we just run a BFS on the subgraph of H induced by the active vertices of V_C and V_D which takes $O((|V_C|+|V_D|)^2) = O(m^{4/5})$ time and get a spanning forest F_{CD}. Due to page limit, some proofs of the following lemmas are omitted and will be given in the full version.

Lemma 3. *The space for storing H is $\tilde{O}(m)$, and it takes $\tilde{O}(m^{6/5})$ time to initialize this structure.*

Lemma 4. *(Consistency of $\bar{E}_T$) For any two active vertices $u \in V \setminus V_A, v \in V_D$, if there is a path longer than one connecting them whose intermediate vertices are all active and in V_A, then for some $T \in F_A$, they are connected by the subset of edges $\bar{E}_T \cup E_A$ induced by the active vertices.*

Proof. From the conditions, all the intermediate vertices on the path between u and v will be in the same spanning tree $T \in F_A$. So if $u \in V_C \cup V_D$, there is an edge connecting u and v in $\bar{E}(T)$. If $u \in V_B$ and $v \in V_D$, by Lemma 1, u will be connected to the representative vertex u_T in E_A, and there is an edge connecting u_T and v directly in $\bar{E}(T)$.

From Lemma 1 and 4, the artificial edges in higher level generated by a spanning tree in a lower level can reflect the connectivity between active higher level vertices through this spanning tree. The subgraph of H induced by a subset will contain all the artificial edges and original edges of G, so it can reflect the connectivity in this subset and lower sets between its active vertices. We have the following lemma, whose proof is omitted due to page limit.

Lemma 5. *For any two active vertices u, v in the set V_i ($0 \le i \le k+1$) or higher, u and v are connected in the subgraph of H induced by the active vertices of $V_i \cup ... \cup V_k \cup V_C \cup V_D$ if and only if they are connected in the subgraph of G induced by the active vertices. Particularly for u, v in $V_C \cup V_D$, u and v are connected in the the subgraph of H induced by the active vertices of $V_C \cup V_D$ if and only if they are connected in the subgraph of G induced by the active vertices.*

3.2 Switching a Vertex

In this section we show how this structure is maintained in $\tilde{O}(m^{4/5})$ time when changing the status of a vertex v. From Lemma 2(4), deleting or inserting an inter-level edge in H may cause changing at most 3 higher inter-level edges in the adjacency graph. However, there are at most $\Theta(\log n)$ vertex sets in this structure, so we need other schemes to bound the number of edges updated during a vertex update. Note that after any vertex update, we will run a BFS on the active vertices of V_C and V_D in $H = G_{k+1}$.

When v is in V_B.

Lemma 6. *The degree of any vertex of V_i in H is at most $(i+1)2^{i+2}m^{1/5}$.*

Lemma 7. *Changing the status of a vertex v in V_i will not affect the lower-level dynamic spanning forests $F_A, F_0, F_1, ..., F_{i-1}$. It can update at most $O(2^i m^{1/5} i \log^3 n) = \tilde{O}(2^i m^{1/5})$ edges in $F_i, F_{i+1}, ..., F_k, F_{CD}$, respectively. Similarly, changing the status of a vertex v in V_A can update at most $\tilde{O}(m^{1/5})$ edges in $F_A, F_0, F_1, ..., F_k, F_{CD}$.*

Proof. Changing the status of a vertex in V_i can only lead to inserting or deleting edges in H associated with a vertex in V_i or higher levels. Thus it will not affect the dynamic spanning forests $F_0, F_1, ..., F_{i-1}$. There are two types of updates:

- Updating v affects a tree $T \in F_i$, so we need to update the list for T in $R(H, F_i, V_{i+1}, ..., V_k, V_C, V_D)$. From Lemma 2(3) and 6, it results in inserting/deleting $O(i \cdot k 2^i m^{1/5})$ edges in $F_{i+1}, ..., F_k, F_{CD}$ or inter-level edges in H.
- From Lemma 2(4), updating an inter-level edge e from a spanning tree T' to a higher level vertex in the previous step will change other edges in H. Here we bound the number of such edges. Let $T' \in F_j$, then in $R(H, F_j, V_{j+1}, ..., V_k, V_C, V_D)$, there are at most $O(k)$ inter-level edges induced by T', and from Note 1, there is only one spanning tree adjacent to T' in every higher level in H. So the number of inter-level edges changed by e is $O(k^2)$. So we need to update $O(2^i m^{1/5} i \log^3 n)$ such inter-level edges. From Lemma 2(4), the total number of edges updated in H is still $\tilde{O}(2^i m^{1/5})$.

Thus, the time needed to update the graph H and the dynamic spanning forests $F_A, F_0, ..., F_k$ when switching a vertex in V_i is equal to $\tilde{O}(2^i m^{1/5})|V_i|^{1/2} = \tilde{O}(2^{(1+i)/2} m^{3/5})$. When $i = k = \lfloor 2/5 \log m \rfloor$, the time bound reaches $\tilde{O}(m^{4/5})$.

When $v \in V_B$, if v is the representative vertex of a tree $T \in F_A$ and v is turned inactive, we need to find another active vertex as the representative for T. If v is turned active and there is no active vertices of V_B associated with a tree $T \in F_A$ adjacent to v, v is chosen as the representative vertex of T. In both cases, we need to find all the vertices in V_D adjacent to T and update $\bar{E}_T$, which takes $O(|V_D|) = O(m^{1/5})$ time using $ET(G, F_A, V_D)$ and Theorem 2. Since v is adjacent to $O(m^{3/5})$ spanning trees in F_A, this procedure takes $O(m^{4/5})$ time.

When v is in V_A. We follow these steps, which also takes $\tilde{O}(m^{4/5})$ time:

- From Lemma 7, changing the status of a vertex v in V_A may update $\tilde{O}(m^{1/5})$ edges in H on $V_B \cup V_C$, and it may update $\tilde{O}(m^{1/5})$ edges in $F_A, F_0, ..., F_k$, F_{CD}, so the time needed for this step is $\tilde{O}(\sqrt{n} m^{1/5}) = \tilde{O}(m^{7/10})$.
- Maintain the structures $ET(G, F_A, V_C)$ and $ET(G, F_A, V_D)$ after updating F_A will take $O(m^{3/5})$ time, because at most $m^{1/5}$ edges will be changed in F_A, and from Theorem 2, every link/cut operation in F_A will take $O(|V_C| + |V_D|) = O(m^{2/5})$ time.
- Consider the edges in $\bar{E}_T$ for a tree $T \in F_A$ connecting V_B and V_D. For all the old spanning trees T of F_A, delete $\bar{E}(T)$ from H. For $\bar{E}_{T'}$ on every new spanning tree T' in F_A after cutting or linking, we find a new active representative vertex in V_B and then construct $\bar{E}_{T'}$. Since there can be at most $m^{1/5}$ link/cut operations in F_A, this may change at most $m^{1/5}|V_D| = O(m^{2/5})$ edges in all the edge sets $\bar{E}_T$ and H.

– Consider all other edges in $\bar{E}_T$ for $T \in F_A$. The number of edges changed in $\bar{E}_T$ when performing a cut/link in F_A is $O((|V_C| + |V_D|)|V_D|) = O(m^{3/5})$. So in fact we need to update $O(m^{4/5})$ such edges in H.

When v is in V_C or V_D. Note that the sets $\bar{E}_T$ do not need to update. If $v \in V_C$, update the structures $R(H, F_i, V_{i+1}, ..., V_k, V_C, V_D)$ $(0 \leq i \leq k)$ and $R(G, F_A, V_0, V_1, ..., V_k, V_C)$ takes $\tilde{O}(m^{4/5})$ time since the degree of v is bounded by $m^{4/5}$. If $v \in V_D$, we still need to update $R(H, F_i, V_{i+1}, ..., V_k, V_C, V_D)$ $(0 \leq i \leq k)$, since the size of V_B is bounded by $2m^{4/5}$, from Lemma 2(1), this will also take $\tilde{O}(m^{4/5})$ time.

Discussion: Why $\tilde{O}(m^{4/5})$ in the worst-case update time? The $O(n^{1/2})$ worst-case edge update connectivity structure [9] is the main bottleneck for our approach. The set of vertices of degrees in the range $[p, q]$ will contain $\leq 2m/p$ vertices, so the vertex update time will be $\tilde{O}(q(m/p)^{1/2}) \geq \tilde{O}(p^{1/2}m^{1/2})$, if we use the edge update structure. However, when p is large enough, we can run a BFS to get connected components after a update, which takes $\tilde{O}(m^2/p^2)$ time. When balancing these two, the update time will be $\tilde{O}(m^{4/5})$. Also, we can get $\tilde{O}(m^{4/5+\epsilon})$ update time and $\tilde{O}(m^{1/5-\epsilon})$ query time by simply changing the degree bound between V_C and V_D to $O(m^{1/5-\epsilon})$.

3.3 Answering a Query

To answer a connectivity query between u and v in the subgraph of G induced by the active vertices, first find the spanning trees $T(u)$ and $T(v)$ in $F_A, F_0, ..., F_k, F_{CD}$ containing u and v, respectively. Then find all higher level spanning trees connecting to $T(u)$ or $T(v)$ and check whether $T(u)$ and $T(v)$ are connected to a common spanning tree in higher levels.

By symmetry, we only discuss finding such spanning trees for u. If $u \in V_A$, we first find $T(u) \in F_A$ which contains u, and then find the spanning trees in $F_0, F_1, ..., F_k, F_{CD}$ which is adjacent to the spanning tree $T(u)$ in H. By Note 1 and Lemma 4, there is only one tree in each forest satisfying this condition. Since we maintain the full active adjacency lists $A_G(T, V_0), ..., A_G(T, V_k), A_G(T, V_C)$ in $R(G, F_A, V_0, V_1, ..., V_k, V_C)$, we can find the trees $T_0, ..., T_k, T_C$ in $F_0, ..., F_k, F_{CD}$ adjacent to $T(u)$ in G in $O(k) = O(\log n)$ time. Those trees are also the ones adjacent to T in H. To find spanning trees in F_{CD} adjacent to T that only contain active vertices in V_D, we need to check whether u' is adjacent to T for all active $u' \in V_D$ by $ET(G, F_A, V_D)$, which takes $O(m^{1/5})$ time.

For any spanning tree $T_i \in F_i$ we have found in V_B or u itself is in a tree T_i of V_B, we recursively run this procedure and find all the trees in $F_{i+1}, ..., F_k, F_{CD}$ connecting to T_i in H, this will take $O(\log n)$ time. Since u can only be connected to one spanning tree in a higher level forest, the time for all T_i will be $O(\log^2 n)$. After this, we check whether there is a common tree in the set of trees connecting to u and v that we found. The running time for the query algorithm is $\tilde{O}(m^{1/5})$.

The correctness of this query algorithm is easy to see from Lemma 5. If we find a common tree connecting to u and v, then u, v must be connected. If u, v are connected in the dynamic G, let w be the highest vertex on the path connecting

u, v, then u, v will be connected to the tree containing w in the subgraph without higher vertices than w, so we have found such spanning tree in our procedure. A complete proof of the correctness will be given in the full version.

4 Dynamic Subgraph Connectivity with $\tilde{O}(m^{2/3})$ Amortized Update Time and Linear Space

In this section, we briefly describe a dynamic subgraph connectivity structure of $\tilde{O}(m^{2/3})$ amortized update time and $\tilde{O}(m^{1/3})$ query time, which improves the structure by Chan, Pătraşcu and Roditty [2] from $\tilde{O}(m^{4/3})$ space to linear space.

Theorem 4. *There is a dynamic subgraph connectivity structure for a graph G with $\tilde{O}(m^{2/3})$ amortized vertex update time, $\tilde{O}(m^{1/3})$ query time, $O(m)$ space and $\tilde{O}(m^{4/3})$ preprocessing time.*

As before, define the subsets of vertices in V by their degrees:

- V_L: vertices of degrees at most $m^{2/3}$.
- V_H: vertices of degrees larger than $m^{2/3}$. So $|V_H| < 2m^{1/3}$.

As in [2], we divide the updates into phases, each consisting of $m^{2/3}$ updates. The active vertices in V_L will be divided into two sets P and Q, where P only undergoes deletions and Q accepts both insertions and deletions. At the beginning of each phase, P contains all the active vertices in V_L and Q is empty. So when a vertex of V_L is turned active, we add it to Q. At the end of that phase, we move all the vertices of Q to P and reinitialize the structure. So the size of Q is bounded by $m^{2/3}$. We also define the set $\bar{Q}$ to be all the vertices that have once been in Q within the current phase, so $|Q| \leq |\bar{Q}| \leq m^{2/3}$. Notice that P and Q only contain active vertices but V_H and $\bar{Q}$ may contain both active and inactive vertices. Then we maintain the following structures for each phase:

- Keep a dynamic spanning forest F in the subgraph of G induced by P which supports edge deletions in polylogarithmic amortized time. [22]
- Maintain the active adjacency structure $E_Q = R(G, F, Q)$.
- Maintain the structure $ET(G, F, V_H)$.
- For every edge $e = (u, v)$ where $u \in P$ and $v \in \bar{Q} \cup V_H$ within the current phase, let T be the spanning tree of F containing u. Then for every vertex w in V_H adjacent to T, we add an edge (v, w) into the set E_H. Since $E_H \in (\bar{Q} \cup V_H) \times V_H$, we just need $O(m)$ space to store E_H.
- Construct a dynamic graph G' containing all the active vertices of $Q \cup V_H$, and all the edges in $E \cup E_Q \cup E_H$ connecting two such vertices. So the number of vertices in G' is $O(m^{2/3})$. Maintain a dynamic spanning forest F' of G' which supports insertions and deletions of edges in polylogarithmic amortized time. [22]

We can see both E_Q and E_H take linear space to store, and from Theorem 2 and the dynamic structure for edge updates [22], the total space is still linear. It takes linear time to initialize $F, E_Q, ET(G, F, V_H), G'$ and $\tilde{O}(m^{4/3})$ time to initialize E_H. To see the consistency of this structure, we have the following lemma, whose proof will be given in the full version of this paper.

Lemma 8. *Two active vertices of $Q \cup V_H$ are connected in G' if and only if they are connected in the subgraph of G induced by the active vertices.*

When updating a vertex in V_L, we analyze the update time by structures:

Maintaining F and E_Q. When deleting a vertex from P, we may split a spanning tree of F into at most $m^{2/3}$ subtrees. So it takes $\tilde{O}(m^{2/3})$ time to maintain E_Q. When updating a vertex in Q, we need to update $O(m^{2/3})$ edges of E_Q from Lemma 2. So it takes $\tilde{O}(m^{2/3})$ time to update F'.

Maintaining E_H. We need to update E_H when a new vertex is inserted to $\bar{Q}$ or when a vertex is deleted from P. When a new vertex is inserted to $\bar{Q}$, we check all the edges associated with it and find the spanning trees in F adjacent to it, then update E_H. When deleting a vertex of P, we find the vertices of V_H adjacent to T which contains that vertex and delete all the outdated edges of E_H. It is hard to bound the time for updating E_H within one update, so we consider the total time needed in one phase. For every edge $e = (u, v)$ where $u \in P$, when v appears in $\bar{Q}$ or V_H, we add $O(m^{1/3})$ edges (v, w) to E_H where w is in V_H and adjacent to the spanning tree in F containing u. As long as u is still in P, the number of such w in V_H can only decrease since P supports deletion only. So only deletions will take place for the edges in E_H induced by e. Thus, updating E_H and the corresponding F' will take $\tilde{O}(m^{4/3})$ time per phase, so we get $\tilde{O}(m^{2/3})$ amortized time.

Maintaining $ET(G, F, V_H)$. By the same reasoning, maintaining the structure $ET(G, F, V_H)$ for one vertex in V_H within one phase will take $\tilde{O}(m)$ time.

Updating a vertex in V_H. We only need to maintain the graph G' when updating a vertex in V_H, which will take $\tilde{O}(m^{2/3})$ time.

Answering a query of connectivity between u and v. If both are in $Q \cup V_H$, by Lemma 8, we check whether they are connected in G'. Otherwise suppose $u \in P$ (or v), we need to find an active vertex u' (or v') in $Q \cup V_H$ which is adjacent to the spanning tree $T \in F$ containing u (v). Similarly to the worst-case structure, we need to check all the active vertices in V_H whether they are adjacent to T by $ET(G, F, V_H)$, which takes $O(m^{1/3})$ time. Thus when only $u \in P$, u, v are connected iff u' and v are connected in G' since the path must go through a vertex in G'. When both of them are in P, they are connected in G iff they are in the same tree of F or u' and v' are connected in G'.

References

1. Chan, T.: Dynamic subgraph connectivity with geometric applications. SIAM J. Comput. 36(3), 681–694 (2006)
2. Chan, T.M., Pătraşcu, M., Roditty, L.: Dynamic connectivity: Connecting to networks and geometry. In: Proceedings 49th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 95–104 (2008)
3. Coppersmith, D., Winograd, T.: Matrix multiplication via arithmetic progressions. In: Proc. 19th ACM Symp. on the Theory of Computing (STOC), pp. 1–6 (1987)

4. Demetrescu, C., Italiano, G.F.: A new approach to dynamic all pairs shortest paths. J. ACM 51(6), 968–992 (2004)
5. Demetrescu, C., Thorup, M., Chowdhury, R.A., Ramachandran, V.: Oracles for distances avoiding a failed node or link. SIAM J. Comput. 37(5), 1299–1318 (2008)
6. Demetrescu, C., Italiano, G.F.: Trade-offs for fully dynamic transitive closure on dags: breaking through the o(n2 barrier. J. ACM 52(2), 147–156 (2005)
7. Duan, R., Pettie, S.: Dual-failure distance and connectivity oracles. In: Proceedings 20th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 506–515 (2009)
8. Duan, R., Pettie, S.: Connectivity oracles for failure prone graphs. In: Proceedings 42nd Annual ACM Symposium on Theory of Computing, STOC (to appear, 2010)
9. Eppstein, D., Galil, Z., Italiano, G., Nissenzweig, A.: Sparsification – a technique for speeding up dynamic graph algorithms. J. ACM 44(5), 669–696 (1997)
10. Frederickson, G.: Data structures for on-line updating of minimum spanning trees, with applications. SIAM J. Comput. 14(4), 781–798 (1985)
11. Frigioni, D., Italiano, G.F.: Dynamically switching vertices in planar graphs. Algorithmica 28(1), 76–103 (2000)
12. Henzinger, M., King, V.: Randomized fully dynamic graph algorithms with polylogarithmic time per operation. J. ACM 46(4), 502–516 (1999)
13. Holm, J., de Lichtenberg, K., Thorup, M.: Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. J. ACM 48(4), 723–760 (2001)
14. King, V.: Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs. In: FOCS 1999: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, Washington, DC, USA. IEEE Computer Society, Los Alamitos (1999)
15. King, V., Sagert, G.: A fully dynamic algorithm for maintaining the transitive closure. In: STOC 1999: Proceedings of the thirty-first annual ACM symposium on Theory of computing, pp. 492–498. ACM, New York (1999)
16. Pătraşcu, M., Thorup, M.: Planning for fast connectivity updates. In: Proceedings 48th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 263–271 (2007)
17. Roditty, L., Zwick, U.: A fully dynamic reachability algorithm for directed graphs with an almost linear update time. In: Proceedings 36th ACM Symposium on Theory of Computing (STOC), pp. 184–191 (2004)
18. Roditty, L., Zwick, U.: On dynamic shortest paths problems. In: Albers, S., Radzik, T. (eds.) ESA 2004. LNCS, vol. 3221, pp. 580–591. Springer, Heidelberg (2004)
19. Roditty, L., Zwick, U.: Improved dynamic reachability algorithms for directed graphs. SIAM J. Comput. 37(5), 1455–1471 (2008)
20. Sankowski, P.: Dynamic transitive closure via dynamic matrix inverse. In: Proceedings 45th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 509–517 (2004)
21. Thorup, M.: Decremental dynamic connectivity. J. Algor. 33(2), 229–243 (1999)
22. Thorup, M.: Near-optimal fully-dynamic graph connectivity. In: Proceedings 32nd ACM Symposium on Theory of Computing (STOC), pp. 343–350 (2000)
23. Thorup, M.: Worst-case update times for fully-dynamic all-pairs shortest paths. In: Proceedings 37th ACM Symposium on Theory of Computing (STOC), pp. 112–119 (2005)

Tight Thresholds for Cuckoo Hashing via XORSAT

(Extended Abstract)

Martin Dietzfelbinger[1,⋆], Andreas Goerdt[2], Michael Mitzenmacher[3,⋆⋆], Andrea Montanari[4,⋆⋆], Rasmus Pagh[5], and Michael Rink[1,⋆]

[1] Fakultät für Informatik und Automatisierung, Technische Universität Ilmenau
{martin.dietzfelbinger,michael.rink}@tu-ilmenau.de
[2] Fakultät für Informatik, Technische Universität Chemnitz
goerdt@informatik.tu-chemnitz.de
[3] School of Engineering and Applied Sciences, Harvard University
michaelm@eecs.harvard.edu
[4] Electrical Engineering and Statistics Departments, Stanford University
montanar@stanford.edu
[5] Efficient Computation group, IT University of Copenhagen
pagh@itu.dk

Abstract. We settle the question of tight thresholds for offline cuckoo hashing. The problem can be stated as follows: we have n keys to be hashed into m buckets each capable of holding a single key. Each key has $k \geq 3$ (distinct) associated buckets chosen uniformly at random and independently of the choices of other keys. A hash table can be constructed successfully if each key can be placed into one of its buckets. We seek thresholds c_k such that, as n goes to infinity, if $n/m \leq c$ for some $c < c_k$ then a hash table can be constructed successfully with high probability, and if $n/m \geq c$ for some $c > c_k$ a hash table cannot be constructed successfully with high probability. Here we are considering the offline version of the problem, where all keys and hash values are given, so the problem is equivalent to previous models of multiple-choice hashing. We find the thresholds for all values of $k > 2$ by showing that they are in fact the same as the previously known thresholds for the random k-XORSAT problem. We then extend these results to the setting where keys can have differing number of choices, and make a conjecture (based on experimental observations) that extends our result to cuckoo hash tables storing multiple keys in a bucket.

1 Introduction

Consider a hashing scheme with n keys to be hashed into m buckets each capable of holding a single key. Each key has $k \geq 3$ (distinct) associated buckets chosen uniformly at random and independently of the choices of other keys. A hash

⋆ Research supported by DFG grant DI 412/10-1.
⋆⋆ Part of this work was done while visiting Microsoft Research New England.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 213–225, 2010.

table can be constructed successfully if each key can be placed into one of its buckets. This setting describes the offline load balancing problem corresponding to multiple choice hashing [1] and cuckoo hashing [12,25] with $k \geq 3$ choices. An open question in the literature (see, for example, the discussion in [23]) is to determine a tight threshold c_k such that if $n/m \leq c$ for some $c < c_k$ then a hash table can be constructed successfully with high probability (whp), and if $n/m \geq c$ for some $c > c_k$ a hash table cannot be constructed successfully whp. In this paper, we provide these thresholds.

We note that, in parallel with this work, other papers have similarly provided means for determining the thresholds [14,13,15]. Our work differs from these works in substantial ways. Perhaps the most substantial is our argument that, somewhat surprisingly, the thresholds we seek were actually essentially already known. We show that tight thresholds follow from known results in the literature, and in fact correspond exactly to the known thresholds for the random k-XORSAT problem. We describe the k-XORSAT problem and the means for computing its thresholds in more detail in the following sections. Our argument is somewhat indirect, although all of the arguments appear to rely intrinsically on the analysis of corresponding random hypergraphs, and hence the alternative arguments of [14,13,15] provide additional insight that may prove useful in further explorations.

With this starting point, we extend our study of the cuckoo hashing problem in two ways. First, we consider *irregular cuckoo hashing*, where the number of choices corresponding to a key is not a fixed constant k but itself a random variable depending on the key. Our motivations for studying this variant include past work on irregular low-density parity-check codes [19] and recent work on alternative hashing schemes that have been said to behave like cuckoo hashing with "3.5 choices" [18]. Beyond finding thresholds, we show how to optimize irregular cuckoo hashing schemes with a specified average number of choices per key; for example, with an average of 3.5 choices per key, the optimal scheme is the natural one where half of the keys obtain 3 choices, and the other half obtain 4. Second, we consider the generalization to the setting where a bucket can hold more than one key. We provide a conjecture regarding the appropriate threshold behavior for this setting. The conjecture is backed by a simple algorithm adapted from Sanders' "selfless algorithm" [26,3] that can be found in the full version of this paper [8, preliminary full version]. Experimentally, it appears to perform remarkably close to the thresholds predicted by our conjecture. After this paper was submitted, we learned that, for sufficiently large bucket sizes, the conjecture was proved in the recent paper [16].

2 Technical Background on Cores

The key to our analysis will be the behavior of cores in random hypergraphs. We therefore begin by providing a review of this subject. To be clear, the results of this section are not new. Readers familiar with the subject may want to

skip this section; other readers are encouraged to see [22, Ch. 18], as well as references [5,10,24] for more background.

We consider the set of all k-uniform hypergraphs with m nodes and n hyperedges $\mathcal{G}^k_{m,n}$. More precisely, each hypergraph G from $\mathcal{G}^k_{m,n}$ consists of n (labeled) hyperedges of a fixed size $k \geq 2$, chosen independently at random, with repetition, from the $\binom{m}{k}$ subsets of $\{1, \ldots, m\}$ of size k. This model will be regarded as a probability space. We always assume k is fixed, m is sufficiently large, and $n = cm$ for a constant c.

For $\ell \geq 2$, the *ℓ-core* of a hypergraph G is defined as the largest induced sub-hypergraph that has minimum degree ℓ or larger. It is well known that the ℓ-core can be obtained by the following iterative "peeling process": While there are nodes with degree smaller than ℓ, delete them and their incident hyperedges. By pursuing this process backwards one sees that the ℓ-core, conditioned on the number of nodes and hyperedges it contains, is a uniform random hypergraph that satisfies the degree constraint.

The fate of a fixed node a after a fixed number of h iterations of the peeling procedure is determined by the h-neighborhood of a, where the h-neighborhood of a is the sub-hypergraph induced on the nodes at distance at most h from a. For example, the 1-neighborhood contains all hyperedges containing a. In our setting where n is linear in m the h-neighborhood of node a is a hypertree of low degree (at most $\log \log m$) whp. We assume this in the discussion to come.

We can see whether a node a is removed from the hypergraph in the course of h iterations of the peeling process in the following way. Consider the hypertree rooted from a (so the children are nodes that share a hyperedge with a, and similarly the children of a node share a hyperedge with that node down the tree). First, consider the nodes at distance $h-1$ from a and delete them if they have at most $\ell-2$ child hyperedges; that is, their degree is at most $\ell-1$. Second, treat the nodes at distance $h-2$ in the same way, and so on, down to distance 1, the children of a. Finally, a is deleted if its degree is at most $\ell - 1$.

The analysis of such random processes on trees has been well-studied in the literature. We wish to determine the probability q_h that node a is deleted after h rounds of the peeling process. For $j < h$ let p_j be the probability that a node at distance $h - j$ from a is deleted after j rounds of the peeling process. The discussion becomes easier for the binomial random hypergraph with an expected number of cm hyperedges: Each hyperedge is present with probability $k! \cdot c/m^{k-1}$ independently. It is well known that $\mathcal{G}^k_{m,n}$ and the binomial hypergraph are equivalent as far as asymptotic behavior of cores are concerned when c is a constant.

Let $\mathrm{Bin}(N, p)$ denote a random variable with a binomial distribution, and $\mathrm{Po}(\beta)$ a random variable with a Poisson distribution. Below we make use of the Poisson approximation of the binomial distribution and the fact that the number of child hyperedges of a node in the hypertree asymptotically follows the binomial distribution. This results in additive terms that tend to zero as m goes to infinity. We have $p_0 = 0$,

$$p_1 = \Pr\left[\mathrm{Bin}\left(\binom{m-1}{k-1},\ k! \cdot \frac{c}{m^{k-1}}\right) \le \ell - 2\right]$$

$$= \Pr[\mathrm{Po}(kc) \le \ell - 2] \pm o(1),$$

$$p_{j+1} = \Pr\left[\mathrm{Bin}\left(\binom{m-1}{k-1},\ k! \cdot \frac{c}{m^{k-1}} \cdot (1-p_j)^{k-1}\right) \le \ell - 2\right]$$

$$= \Pr[\mathrm{Po}(kc(1-p_j)^{k-1}) \le \ell - 2] \pm o(1), \text{ for } j = 1, \ldots, h-2.$$

The probability q_h that a itself is deleted is then instead given by

$$q_h = \Pr[\mathrm{Po}(kc(1-p_{h-1})^{k-1}) \le \ell - 1] \pm o(1). \tag{1}$$

The p_j are monotonically increasing and $0 \le p_j \le 1$, so $p = \lim p_j$ is well-defined. The probability that a is deleted approaches p from below as h grows. Continuity implies that p is the smallest non-negative solution of

$$p = \Pr[\mathrm{Po}(kc(1-p)^{k-1}) \le \ell - 2].$$

Observe that 1 is always a solution. Equivalently, applying the monotone function $t \mapsto kc(1-t)^{k-1}$ to both sides of the equation, p is the smallest solution of

$$kc(1-p)^{k-1} = kc\left(1 - \Pr[\mathrm{Po}(kc(1-p)^{k-1}) \le \ell - 2]\right)^{k-1}. \tag{2}$$

Let $\beta = kc(1-p)^{k-1}$. It is helpful to think of β with the following interpretation: Given a node in the hypertree, the number of child hyperedges (before deletion) follows the distribution $\mathrm{Po}(kc)$. Asymptotically, a given child hyperedge is not deleted with probability $(1-p)^{k-1}$, independently for all children. Hence the number of child hyperedges after deletion follows the distribution $\mathrm{Po}(kc(1-p)^{k-1})$. Hence β is the key parameter giving the expected number of hyperedges containing a node that could contribute to keeping it in the core.

Note that (2) is equivalent to

$$c = \frac{1}{k} \cdot \frac{\beta}{(\Pr[\mathrm{Po}(\beta) \ge \ell - 1])^{k-1}}.$$

This motivates considering the function

$$g_{k,\ell}(\beta) = \frac{1}{k} \cdot \frac{\beta}{(\Pr[\mathrm{Po}(\beta) \ge \ell - 1])^{k-1}}, \tag{3}$$

which has the following properties in the range $(0, \infty)$: It tends to infinity for $\beta \to 0$, as well as for $\beta \to \infty$. Since it is convex there is exactly one global minimum. Let $\beta^*_{k,\ell} = \arg\min_\beta g_{k,\ell}(\beta)$ and $c^*_{k,\ell} = \min g_{k,\ell}(\beta)$. For $\beta > \beta^*_{k,\ell}$ the function $g_{k,\ell}$ is monotonically increasing. For each $c > c^*_{k,\ell}$ let $\beta(c) = \beta_{k,\ell}(c)$ denote the unique $\beta > \beta^*_{k,\ell}$ such that $g_{k,\ell}(\beta) = c$.

Coming back to the fate of a under the peeling process, Equation (1) shows that a is deleted with probability approaching $\Pr[\mathrm{Po}(\beta(c)) \le \ell - 1]$. This probability is smaller than 1 if and only if $c > c^*_{k,\ell}$, which implies that the expected number of nodes that are *not* deleted is linear in n. As the h-neighborhoods of two nodes a and b are disjoint whp, by making use of the second moment we can show that in this case a linear number of nodes survive whp. (The sophisticated reader would use Azuma's inequality to obtain concentration bounds.)

Following this line of reasoning, we obtain the following results, the full proof of which is in [24]. (See also the related argument of [22, Ch. 18].) Note the restriction to the case $k + \ell > 4$, which means that the result does not apply to 2-cores in standard graphs; since the analysis of standard cuckoo hashing is simple, using direct arguments, this case is ignored in the analysis henceforth.

Proposition 1. *Let $k + \ell > 4$ and G be a random hypergraph from $\mathcal{G}^k_{m,n}$. Then $c^*_{k,\ell}$ is the threshold for the* appearance *of an ℓ-core in G. That is, for constant c and $m \to \infty$,*
(a) *if $n/m = c < c^*_{k,\ell}$, then G has an empty ℓ-core with probability $1 - o(1)$.*
(b) *if $n/m = c > c^*_{k,\ell}$, then G has an ℓ-core of linear size with probability $1 - o(1)$.*

In the following we assume $c > c^*_{k,\ell}$. Therefore $\beta(c) > \beta^*_{k,\ell}$ exists. Let $\hat{m}$ be the number of nodes in the ℓ-core and $\hat{n}$ be the number of hyperedges in the ℓ-core. We find it useful in what follows to consider the *edge density* of the ℓ-core, which is the ratio of the number of hyperedges to the number of nodes.

Proposition 2. *Let $c > c^*_{k,\ell}$ and $n/m = c\,(1 \pm o(1))$. Then whp in $\mathcal{G}^k_{m,n}$*

$$\hat{m} = \Pr[\mathrm{Po}(\beta(c)) \ge \ell] \cdot m \pm o(m) \text{ and } \hat{n} = (\Pr[\mathrm{Po}(\beta(c)) \ge \ell - 1])^k \cdot n \pm o(m).$$

The bound for $\hat{m}$ follows from the concentration of the expected number of nodes surviving when we plug in the limit p for p_h in equation (1). The result for $\hat{n}$ follows similar lines: Consider a fixed hyperedge e that we assume is present in the random hypergraph. For each node of this hyperedge we consider its h-neighborhood modified in that e itself does not belong to this h-neighborhood. We have k disjoint trees whp. Therefore each of the k nodes of e survives h iterations of the peeling procedure independently with probability $\Pr[\mathrm{Po}(\beta(c)) \ge \ell - 1]$. Note that we use $\ell - 1$ here (instead of ℓ) because the nodes belong to e. Then e itself survives with $(\Pr[\mathrm{Po}(\beta(c)) \ge \ell - 1])^k$. Concentration of the number of surviving hyperedges again follows from second moment calculations or Azuma's inequality.

Proposition 3. *If $c > c^*_{k,\ell}$ and $n/m = c\,(1 \pm o(1))$ then whp the edge density of the ℓ-core of a random hypergraph from $\mathcal{G}^k_{m,n}$ is*

$$\frac{\beta(c) \cdot \Pr[\mathrm{Po}(\beta(c)) \ge \ell - 1]}{k \cdot \Pr[\mathrm{Po}(\beta(c)) \ge \ell]} \pm o(1).$$

This follows directly from Proposition 2, where we have also used equation (3) to simplify the expression for $\hat{n}$.

We define $c_{k,\ell}$ as the unique c that satisfies

$$\frac{\beta(c) \cdot \Pr[\mathrm{Po}(\beta(c)) \geq \ell - 1]}{k \cdot \Pr[\mathrm{Po}(\beta(c)) \geq \ell]} = \ell - 1. \tag{4}$$

The values $c_{k,\ell}$ will prove important in the work to come; in particular, we next show that $c_{k,2}$ is the threshold for k-ary cuckoo hashing for $k > 2$. We also conjecture that $c_{k,\ell+1}$ is the threshold for k-ary cuckoo hashing when a bucket can hold ℓ keys instead of a single key.

The following table contains numerical values of $c_{k,\ell}$ for $\ell = 2, \ldots, 7$ and $k = 2, \ldots, 6$ (rounded to 10 decimal places). Some of these numbers are found or referred to in other works, such as [5, Sect. 5], [21, Sect. 4.4], [22, p. 423], [11], and [3].

$\ell \backslash k$	2	3	4	5	6
2	–	0.9179352767	0.9767701649	0.9924383913	0.9973795528
3	1.7940237365	1.9764028279	1.9964829679	1.9994487201	1.9999137473
4	2.8774628058	2.9918572178	2.9993854302	2.9999554360	2.9999969384
5	3.9214790971	3.9970126256	3.9998882644	3.9999962949	3.9999998884
6	4.9477568093	4.9988732941	4.9999793407	4.9999996871	4.9999999959
7	5.9644362395	5.9995688805	5.9999961417	5.9999999733	5.9999999998

3 Equality of Thresholds for Random k-XORSAT and k-Ary Cuckoo Hashing

We now recall the random k-XORSAT problem and describe its relationship to cores of random hypergraphs and cuckoo hashing. The k-XORSAT problem is a variant of the satisfiability problem in which every clause has k literals and the clause is satisfied if the XOR of values of the literals is 1. Equivalently, since XORs correspond to addition modulo 2, and the negation of X_i is just 1 XOR X_i, an instance of the k-XORSAT problem corresponds to a system of linear equations modulo 2, with each equation having k variables, and randomly chosen right hand sides. (In what follows we simply use the addition operator where it is understood we are working modulo 2 from context.)

For a random k-XORSAT problem, let $\Phi^k_{m,n}$ be the set of all sequences of n linear equations over m variables $x_1, \ldots, x_m$, where an equation is

$$x_{j_1} + \cdots + x_{j_k} = b_j,$$

where $b_j \in \{0, 1\}$ and $\{j_1, \ldots, j_k\}$ is a subset of $\{1, \ldots, m\}$ with k elements. We consider $\Phi^k_{m,n}$ as a probability space with the uniform distribution.

Given a k-XORSAT formula F, it is clear that F is satisfiable if and only if the formula obtained from F by repeatedly deleting variables that occur only once (and equations containing them) is satisfiable. Now consider the k-XORSAT formula as a hypergraph, with nodes representing variables and hyperedges representing equations. (The values b_j of the equations are not represented.) The

process of repeatedly deleting all variables that occur only once, and the corresponding equations, is exactly equivalent to the peeling process on the hypergraph. After this peeling process, we obtain the 2-core of the hypergraph.

This motivates the following definition. Let $\Psi^k_{m,n}$ be the set of all sequences of n equations such that each variable appears at least twice. We consider $\Psi^k_{m,n}$ as a probability space with the uniform distribution.

From the corresponding fact for hypergraphs it follows that if we start with a uniformly chosen random k-XORSAT formula and perform the peeling process, then conditioned on the remaining number of equations and variables ($\hat{n}$ and $\hat{m}$), we are in fact left with a uniform random formula from $\Psi^k_{\hat{m},\hat{n}}$. Hence, the imperative question is when a random formula from $\Psi^k_{\hat{m},\hat{n}}$ will be satisfiable. In [10], it was shown that this depends entirely on the edge density of the corresponding hypergraph. If the edge density is smaller than 1, so that there are more variables than equations, the formula is likely to be satisfiable, and naturally, if there are more equations than variables, the formula is likely to be unsatisfiable. Specifically, we have the following theorem from [10].

Theorem 1. *Let $k > 2$ be fixed. For $n/m = \gamma$ and $m \to \infty$,*
(a) *if $\gamma > 1$ then a random formula from $\Psi^k_{m,n}$ is unsatisfiable whp.*
(b) *if $\gamma < 1$ then a random formula from $\Psi^k_{m,n}$ is satisfiable whp.*

The proof of Theorem 1 in Section 3 of [10] uses a first moment method argument for the simple direction (part (a)). Part (b) is significantly more complicated, and is based on the second moment method. Essentially the same problem has also arisen in coding theoretic settings; analysis and techniques can be found in for example [20]. It has been suggested by various readers of earlier drafts of this paper that previous proofs of Theorem 1 have been insufficiently complete, particularly for $k > 3$. We therefore provide a detailed proof in [8] for completeness.

We have shown that the edge density is concentrated around a specific value depending on the initial ratio c of hyperedges (equations) to nodes (variables). Let $c_{k,2}$ be the value of c such that the resulting edge density is concentrated around 1. Then Proposition 3 and Theorem 1 together with the preceding consideration implies:

Corollary 1. *Let $k > 2$ and consider $\Phi^k_{m,n}$. The satisfiability threshold with respect to the edge density $c = n/m$ is $c_{k,2}$.*

Again, up to this point, everything we have stated was known from previous work. We now provide the connection to cuckoo hashing, to show that we obtain the same threshold values for the success of cuckoo hashing. That is, we argue the following:

Theorem 2. *For $k > 2$, $c_{k,2}$ is the threshold for k-ary cuckoo hashing to work. That is, with n keys to be stored and m buckets, with $c = n/m$ fixed and $m \to \infty$,*
(a) *if $c > c_{k,2}$, then k-ary cuckoo hashing does not work whp.*
(b) *if $c < c_{k,2}$, then k-ary cuckoo hashing works whp.*

Proof: Assume a set S of n keys is given, and for each $x \in S$ a random set $A_x \subseteq \{1, \ldots, m\}$ of size k of possible buckets is chosen.

To prove part (a), note that the sets A_x for $x \in S$ can be represented by a random hypergraph from $\mathcal{G}^k_{m,n}$. If $n/m = c > c_{k,2}$ and $m \to \infty$, then whp the edge density in the 2-core is greater than 1. The hyperedges in the 2-core correspond to a set of keys, and the nodes in the 2-core to the buckets available for these keys. Obviously, then, cuckoo hashing does not work.

To prove part (b), consider the case where $n/m = c < c_{k,2}$ and $m \to \infty$. Picking for each $x \in S$ a random $b_x \in \{0, 1\}$, the sets A_x, $x \in S$, induce a random system of equations from $\Phi^k_{m,n}$. Specifically, $A_x = \{j_1, \ldots, j_k\}$ induces the equation $x_{j_1} + \cdots + x_{j_k} = b_x$.

By Corollary 1 a random system of equations from $\Phi^k_{m,n}$ is satisfiable whp. This implies that the matrix $M = (m_{i,j})_{i,j} \in \{0, 1\}^{n \times m}$ made up from the left-hand sides of these equations consists of linearly independent rows whp. This is because a given set of left-hand sides with dependent rows is only satisfiable with probability at most $1/2$ when we pick the b_x at random.

Therefore M contains an $n \times n$-submatrix M' with nonzero determinant. We consider the Leibniz expansion of $\det(M')$ over $\mathbb{Z}_2$ as a sum of $n!$ (signed) products of the form $p_\sigma = \prod_{i=1}^n m_{i,\sigma(i)}$, where σ is a bijection between $\{1, \ldots, n\}$ and the set of column indices of M'. At least one of the p_σ must be 1, which implies that $m_{i,\sigma(i)} = 1$ and hence $\sigma(i) \in A_{x_i}$, for $1 \le i \le n$. Thus cuckoo hashing works. □

We make some additional remarks. We note that the idea of using the rank of the key-bucket matrix to obtain lower bounds on the cuckoo hashing threshold is not new either; it appears in [9]. There the authors use a result bounding the rank by Calkin [4] to obtain a lower bound on the threshold, but this bound is not tight in this context. More details can be found by reviewing [4, Theorem 1.2] and [22, Exercise 18.6]. Also, Batu et al. [2] note that 2-core thresholds provide an upper bound on the threshold for cuckoo hashing, but fail to note the connection to work on the k-XORSAT problems.

4 Non-integer Choices

The analysis of k-cores in Section 3 and the correspondence to k-XORSAT problems extends nicely to the setting where the number of choices for a key is not necessarily a fixed number k. This can be naturally accomplished in the following way: when a key x is to be inserted in the cuckoo hash table, the number of choices of location for the key is itself determined by some hash function; then the appropriate number of choices for each key x can also be found when performing a lookup. Hence, it is possible to ask about for example cuckoo hashing with 3.5 choices, by which we would mean an average of 3.5 choices. Similarly, even if we decide to have an average of k choices per key, for an integer k, it is not immediately obvious whether the success probability in k-ary cuckoo hashing could be improved if we do not fix the number of possible positions for a key but rather choose it at random from a cleverly selected distribution.

Let us consider a more general setting where for each $x \in U$ the set A_x is chosen uniformly at random from the set of all k_x-element subsets of $[m]$, where k_x follows some probability mass function ρ_x on $\{2, \ldots, m\}$.[1] Let $\kappa_x = E(k_x)$ and $\kappa^* = \frac{1}{n}\sum_{x \in S} \kappa_x$. Note that κ^* is the average (over all $x \in S$) worst case lookup time for successful searches. We keep κ^* fixed and study which sequence $(\rho_x)_{x \in S}$ maximizes the probability that cuckoo hashing is successful.

We fix the sequence of the expected number of choices per key $(\kappa_x)_{x \in S}$ and therefore κ^*. Furthermore we assume $\kappa_x \leq n - 2$, for all $x \in S$; obviously this does not exclude interesting cases. For compactness reasons, there is a system of probability mass functions ρ_x that maximizes the success probability. The proof of the following is given in [8].

Proposition 4. *Let $(\rho_x)_{x \in S}$ be an optimal sequence. Then for all $x \in S$:*

$$\rho_x(\lfloor \kappa_x \rfloor) = 1 - (\kappa_x - \lfloor \kappa_x \rfloor), \text{ and } \rho_x(\lfloor \kappa_x \rfloor + 1) = \kappa_x - \lfloor \kappa_x \rfloor.$$

That is, the success probability is maximized if for each $x \in S$ the number of choices k_x is concentrated on $\lfloor \kappa_x \rfloor$ and $\lfloor \kappa_x \rfloor + 1$ (when the number of choices is non-integral). Further, in the natural case where all keys x have the same expected number κ^* of choices, the optimal assignment is concentrated on $\lfloor \kappa^* \rfloor$ and $\lfloor \kappa^* \rfloor + 1$. Also, if κ_x is an integer, then a fixed degree $k_x = \kappa_x$ is optimal. This is very different from other similar scenarios, such as erasure- and error-correcting codes, where irregular distributions have proven beneficial [19].

We now describe how to extend our previous analysis to derive thresholds for the case of a non-integral number of choices per key; equivalently, we are making use of thresholds for XORSAT problems with an irregular number of literals per clause.

Following notation that is frequently used in the coding literature, we let Λ_k be the probability that a key obtains k choices, and define $\Lambda(x) = \sum_k \Lambda_k x^k$. Clearly, then, $\Lambda'(x) = \sum_k \Lambda_k k x^{k-1}$, and $\Lambda'(1) = \kappa^*$. (We assume henceforth that $\Lambda_0 = \Lambda_1 = 0$ and $\Lambda_k = 0$ for all k sufficiently large for technical convenience.)

We now follow our previous analysis from Section 2; to see if a node a is deleted after h rounds of the peeling process, we let p_j be the probability that a node at distance $h - j$ from a is deleted after j rounds. We must now account for the differing degrees of hyperedges. Here, the appropriate asymptotics is given by a mixture of binomial hypergraphs, with each hyperedge of degree k present with probability $k! \cdot c\Lambda_k / m^{k-1}$ independently.

[1] We could in principle also consider the possibility of keys having only a single choice. However, this is generally not very interesting since even a small number of keys with a single choice would make an assignment impossible whp, by the birthday paradox. Hence, we restrict our attention to at least two choices.

The corresponding equations are then given by $p_0 = 0$,

$$\begin{aligned} p_1 &= \Pr\left[\sum_k \mathrm{Bin}\left(\binom{m-1}{k-1},\ k! \cdot \frac{c\Lambda_k}{m^{k-1}}\right) \le \ell - 2\right] \\ &= \Pr\left[\sum_k \mathrm{Po}(kc\Lambda_k) \le \ell - 2\right] \pm o(1) \\ &= \Pr[\mathrm{Po}(c\Lambda'(1)) \le \ell - 2] \pm o(1), \end{aligned}$$

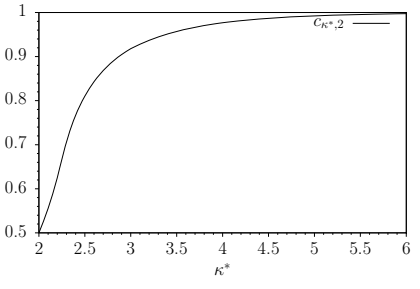
$$p_{j+1} = \Pr\left[\sum_k \mathrm{Bin}\left(\binom{m-1}{k-1},\ k! \cdot \frac{c\Lambda_k}{m^{k-1}} \cdot (1-p_j)^{k-1}\right) \le \ell - 2\right]$$

$$= \Pr\left[\sum_k \mathrm{Po}(kc\Lambda_k(1-p_j)^{k-1}) \le \ell - 2\right] \pm o(1), \text{ for } j = 1, \ldots, h-2$$

$$= \Pr[\mathrm{Po}(c\Lambda'(1-p_j)) \le \ell - 2] \pm o(1), \text{ for } j = 1, \ldots, h-2.$$

Note that we have used the standard fact that the sum of Poisson random variables is itself Poisson, which allows us to conveniently express everything in terms of the generating function $\Lambda(x)$ and its derivative. As before we find $p = \lim p_j$, which is now given by the smallest non-negative solution of

$$p = \Pr[\mathrm{Po}(c\Lambda'(1-p)) \le \ell - 2].$$



κ^*	$c_{\kappa^*,2}$	κ^*	$c_{\kappa^*,2}$
2.25	0.6666666667	4.25	0.9825693463
2.50	0.8103423635	4.50	0.9868637629
2.75	0.8788457372	4.75	0.9900548807
3.00	0.9179352767	5.00	0.9924383913
3.25	0.9408047937	5.25	0.9942189481
3.50	0.9570796377	5.50	0.9955692011
3.75	0.9685811888	5.75	0.9965961383
4.00	0.9767701649	6.00	0.9973795528

Fig. 1. Thresholds for non-integral κ^*-ary cuckoo hashing, with optimal degree distribution. Values in the tables are rounded to the nearest multiple of 10^{-10}.

When given a degree distribution $(\Lambda_k)_k$, we can proceed as before to find the threshold load that allows that the edge density of the 2-core remains greater than 1; using a second moment argument, this can again be shown to be the required property for the corresponding XORSAT problem to have a solution, and hence for there to be a permutation successfully mapping keys to buckets. Details can be found in [8]. Notice that this argument works for all degree distributions (subject to the restrictions given above), but in particular we have

already shown that the optimal thresholds are to be found by the simple degree distributions that have all weight on two values, $\lfloor \kappa^* \rfloor$ and $\lfloor \kappa^* \rfloor + 1$. Abusing notation slightly, let $c_{\kappa^*,2}$ be the unique c such that the edge density of the 2-core of the corresponding mixture is equal to 1, following the same form as in Proposition 3 and equation (4). The corresponding extension to Theorem 2 is the following:

Theorem 3. *For $\kappa^* > 2$, $c_{\kappa^*,2}$ is the threshold for cuckoo hashing with an average of κ^* choices per key to work. That is, with n keys to be stored and m buckets, with $c = n/m$ fixed and $m \to \infty$,*

(a) *if $c > c_{\kappa^*,2}$, for any distribution on the number of choices per key with mean κ^*, cuckoo hashing does not work whp.*

(b) *if $c < c_{\kappa^*,2}$, then cuckoo hashing works whp when the distribution on the number of choices per key is given by $\rho_x(\lfloor \kappa_x \rfloor) = 1 - (\kappa_x - \lfloor \kappa_x \rfloor)$ and $\rho_x(\lfloor \kappa_x \rfloor + 1) = \kappa_x - \lfloor \kappa_x \rfloor$, for all $x \in S$.*

We have determined the thresholds numerically for a range of values of κ^*. The results are shown in Figure 1. One somewhat surprising finding is that the threshold for $\kappa^* \leq 2.25$ appears to simply be given by $c = 0.5/(3 - \kappa^*)$. Consequently, in place of using two hash functions per key, simply by using a mix of two or three hash functions for a key, we can increase the space utilization by adding 33% more keys with the same (asymptotic) amount of memory.

5 A Conjecture and Placement Algorithm

There is as yet no rigorous analysis of the appropriate thresholds for cuckoo hashing for the cases $k > 2$ and $\ell > 1$. However, our results of Section 2 suggest a natural conjecture:

Conjecture 1. For k-ary cuckoo hashing with bucket size ℓ, it is *conjectured* that cuckoo hashing works whp if $n/m = c > c_{k,\ell+1}$, and does not work if $n/m = c < c_{k,\ell+1}$, i. e., that the threshold is at the point where the $(\ell + 1)$-core of the cuckoo hypergraph starts having edge density larger than ℓ.

We have tested this conjecture with a novel algorithm for finding a placement for the keys using k-ary cuckoo hashing when the set S of keys is given in advance. The algorithm is an adaptation of the "selfless algorithm" proposed by Sanders [26], for the case $k = 2$, and analyzed in [3], for orienting standard undirected random graphs so that all edges are directed and the maximum indegree of all nodes is at most ℓ, for some fixed $\ell \geq 2$. We generalize this algorithm to hypergraphs, including hypergraphs where hyperedges can have varying degrees. As we learned after this paper was submitted the conjecture was proved, for sufficiently large bucket sizes, in the recent paper [16].

Of course, maximum matching algorithms can solve this problem perfectly. However, there are multiple motivations for considering our algorithms. First, it seems in preliminary experiments that the running times of standard matching algorithms like the Hopcroft-Karp algorithm [17] will tend to increase significantly as the edge density approaches the threshold (the details of this effect are

not yet understood), while our algorithm has linear running time which does not change in the neighborhood of the threshold. This proves useful in our experimental evaluation of thresholds. Second, we believe that algorithms of this form may prove easier to analyze for some variations of the problem.

Due to space limitations, the description of the algorithm and the experimental results are not presented here, but are given in [8].

6 Conclusion

We have found tight thresholds for cuckoo hashing with 1 key per bucket, by showing that the thresholds are in fact the same for the previously studied k-XORSAT problem. We have generalized the result to irregular cuckoo hashing where keys may have differing numbers of choices, and have conjectured thresholds for the case where buckets have size larger than 1 based on an extrapolation of our results.

References

1. Azar, Y., Broder, A., Karlin, A., Upfal, E.: Balanced allocations. SIAM J. Comput. 29(1), 180–200 (1999)
2. Batu, T., Berenbrink, P., Cooper, C.: Balanced allocations: Balls-into-bins revisited and chains-into-bins, CDAM Research Report Series. LSE-CDAM-2007-34
3. Cain, J.A., Sanders, P., Wormald, N.C.: The random graph threshold for k-orientiability and a fast algorithm for optimal multiple-choice allocation. In: Proc. 18th ACM-SIAM SODA, pp. 469–476 (2007)
4. Calkin, N.J.: Dependent sets of constant weight binary vectors. Combinatorics, Probability, and Computing 6(3), 263–271 (1997)
5. Cooper, C.: The size of the cores of a random graph with a given degree sequence. Random Structures and Algorithms 25(4), 353–375 (2004)
6. Creignou, N., Daudé, H.: Smooth and sharp thresholds for random k-XOR-CNF satisfiability. Theoretical Informatics and Applications 37(2), 127–147 (2003)
7. Creignou, N., Daudé, H.: The SAT-UNSAT transition for random constraint satisfaction problems. Discrete Mathematics 309(8), 2085–2099 (2009)
8. Dietzfelbinger, M., Goerdt, A., Mitzenmacher, M., Montanari, A., Pagh, R., Rink, M.: Tight Thresholds for Cuckoo Hashing via XORSAT. CoRR, abs/0912.0287 (2009)
9. Dietzfelbinger, M., Pagh, R.: Succinct data structures for retrieval and approximate membership. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 385–396. Springer, Heidelberg (2008)
10. Dubois, O., Mandler, J.: The 3-XORSAT threshold. In: Proc. 43rd FOCS, pp. 769–778 (2002)
11. Fernholz, D., Ramachandran, V.: The k-orientability thresholds for $G_{n,p}$. In: Proc. 18th ACM-SIAM SODA, pp. 459–468 (2007)
12. Fotakis, D., Pagh, R., Sanders, P., Spirakis, P.: Space efficient hash tables with worst case constant access time. Theory Comput. Syst. 38(2), 229–248 (2005)

13. Fountoulakis, N., Panagiotou, K.: Orientability of random hypergaphs and the power of multiple choices. In: Abramsky S., et al. (eds.): ICALP 2010, Part I. LNCS, vol. 6198, pp. 348–359. Springer, Heidelberg (2010)
14. Fountoulakis, N., Panagiotou, K.: Sharp load thresholds for cuckoo hashing. CoRR, abs/0910.5147 (2009)
15. Frieze, A.M., Melsted, P.: Maximum matchings in random bipartite graphs and the space utilization of cuckoo hashtables. CoRR, abs/0910.5535 (2009)
16. Gao, P., Wormald, N.C.: Load balancing and orientability thresholds for random hypergraphs. In: 42nd ACM STOC (to appear, 2010)
17. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. 2(4), 225–231 (1973)
18. Lehman, E., Panigrahy, R.: 3.5-way cuckoo hashing for the price of 2-and-a-bit. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 671–681. Springer, Heidelberg (2009)
19. Luby, M., Mitzenmacher, M., Shokrollahi, M.A., Spielman, D.: Efficient erasure correcting codes. IEEE Transactions on Information Theory 47(2), 569–584 (2001)
20. Méasson, C., Montanari, A., Urbanke, R.: Maxwell construction: the hidden bridge between iterative and maximum a posteriori decoding. IEEE Transactions on Information Theory 54(12), 5277–5307 (2008)
21. Mézard, M., Ricci-Tersenghi, F., Zecchina, R.: Two solutions to diluted p-spin models and XORSAT problems. J. Statist. Phys. 111(3/4), 505–533 (2003)
22. Mézard, M., Montanari, A.: Information, Physics, and Computation. Oxford University Press, Oxford (2009)
23. Mitzenmacher, M.: Some open questions related to cuckoo hashing. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 1–10. Springer, Heidelberg (2009)
24. Molloy, M.: Cores in random hypergraphs and Boolean formulas. Random Structures and Algorithms 27(1), 124–135 (2005)
25. Pagh, R., Rodler, F.F.: Cuckoo hashing. J. Algorithms 51(2), 122–144 (2004)
26. Sanders, P.: Algorithms for scalable storage servers. In: Van Emde Boas, P., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2004. LNCS, vol. 2932, pp. 82–101. Springer, Heidelberg (2004)

Resource Oblivious Sorting on Multicores

Richard Cole[1,⋆] and Vijaya Ramachandran[2,⋆⋆]

[1] Computer Science Dept., Courant Institute, NYU, New York, NY 10012
cole@cs.nyu.edu
[2] Dept. of Computer Science, Univ. of Texas, Austin, TX 78712
vlr@cs.utexas.edu

Abstract. We present a new deterministic sorting algorithm that interleaves the partitioning of a sample sort with merging. Sequentially, it sorts n elements in $O(n \log n)$ time cache-obliviously with an optimal number of cache misses. The parallel complexity (or critical path length) of the algorithm is $O(\log n \log\log n)$, which improves on previous bounds for deterministic sample sort. Given a multicore computing environment with a global shared memory and p cores, each having a cache of size M organized in blocks of size B, our algorithm can be scheduled effectively on these p cores in a cache-oblivious manner.

We improve on the above cache-oblivious processor-aware parallel implementation by using the Priority Work Stealing Scheduler (PWS) that we presented recently in a companion paper [12]. The PWS scheduler is both processor- and cache-oblivious (i.e., resource oblivious), and it tolerates asynchrony among the cores. Using PWS, we obtain a resource oblivious scheduling of our sorting algorithm that matches the performance of the processor-aware version. Our analysis includes the delay incurred by false-sharing. We also establish good bounds for our algorithm with the randomized work stealing scheduler.

1 Introduction

We present a new parallel sorting algorithm, which we call *Sample, Partition, and Merge Sort (SPMS)*. It has a critical path length of $O(\log n \log\log n)$ and performs optimal $O(n \log n)$ operations with optimal sequential cache misses. More importantly, using the PWS scheduler for multicores developed and analyzed in [12], and new algorithmic techniques given in this paper, we can schedule it resource-obliviously on a multicore while maintaining these performance bounds. We present background information on multicores, cache-efficiency and resource-obliviousness in Section 2.

The core of the sorting algorithm is a recursive multi-way merging procedure. A notable and novel aspect of this procedure is that it creates its recursive subproblems using a sample sort methodology. We view the sorting algorithm as interleaving a merge sort with a sample sort in a natural way.

⋆ This work was supported in part by NSF Grant CCF-0830516.
⋆⋆ This work was supported in part by NSF Grant CCF-0850775 and CCF-0830737.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 226–237, 2010.

Previous Work. Sorting is a fundamental algorithmic problem, and has been studied extensively. For our purposes, the most relevant results are sequential cache-oblivious sorting, for which provably optimal algorithms are known [15], optimal sorting algorithms addressing pure parallelism [3, 11], and recent work on multicore sorting [5, 4, 6, 16].

The existing multicore algorithms take two main approaches. The first is merge sort [4, 6, 5], either simple or the pipelined method from [11]. The second is deterministic sampling [16]: this approach splits the input into subsets, sorts the subsets, samples the sorted subsets, sort the sample, partitions about a subsample, and recursively sorts the resulting sets. Our algorithm can be viewed as applying this approach to the problem of merging a *suitable number* of sorted sets, which eliminates the need for the first two steps, resulting in significant speed-up.

More specifically, the algorithm in [6] is a simple multicore mergesort; it has polylog parallel time, and good, though not optimal cache efficiency; it is cache-oblivious for private caches (the model we consider in this paper). The algorithm in [4] achieves the optimal caching bound on an input of length n, with $O(\log n)$ parallel time (modulo dependence on cache parameters), but it is both cache-aware and core-aware; this algorithm is based on [11]. The algorithm in [5] is cache oblivious with $O(\log^2 n)$ parallel time, but due to an additive term the cache performance is not optimal on a multicore. The algorithm in [16] is designed for a BSP-style version of a cache aware, multi-level multicore. It uses a different collection of parameters, and so it is difficult to compare with it directly.

Roadmap. In Section 2 we present some background on multicores, and then state our main sorting result. In Section 3 we give a high level description of our parallel sorting algorithm, omitting the details needed to have a resource oblivious implementation. In Section 4, we review the computation model, the work stealing scheduler PWS, and the class of BP algorithms, as developed in [12]. In Section 5, we return to the sorting algorithm, describing the details needed for a resource oblivious implementation, and this is followed by its analysis. Due to space constraints, our matching lower bound for cache optimality (adapted from [2]), along with most of the proofs, are deferred to the full paper [13].

2 Statement of Our Results

Before stating our main result, we give some background, as developed in [12].

Multicore with Private Caches. We model a multicore as consisting of p *cores* (or processors) with an arbitrarily large main memory, which serves as a shared memory. Additionally, each core has a private cache of size M. Data in the main memory is organized in blocks of size B, and the initial input of size n is in main memory, in n/B blocks. When a core C needs a data item x that is not in its private cache, it reads in the block β that contains x from main memory. This new block replaces an existing block in the private cache, which is evicted using an optimal cache replacement policy (LRU suffices for our algorithms, but we do not elaborate further). If another core C' modifies an entry in β, then β is

invalidated in C's cache, and the next time core C needs to access data in block β, an updated copy of β is brought into C's cache.

Cache and Block Misses. We distinguish between two types of cache-related costs incurred in a parallel execution.

The term *cache miss* denotes a read of a block from shared-memory into core C's cache, when a needed data item is not currently in the cache, either because the block was never read by core C, or because it was evicted from C's cache to make room for new data. This is the standard type of cache miss that occurs, and is accounted for, in sequential cache complexity analysis.

The term *block miss* denotes an update by a core $C' \neq C$ to an entry in a block β that is in core C's cache; this entails core C' acquiring block β; if C has a subsequent write, it needs to reacquire the block. This type of 'cache miss' does not occur in a sequential computation, and is a problematic situation that can occur quite often, especially in the resource oblivious setting that we seek.

Resource Obliviousness. We have claimed that our multicore algorithms are resource oblivious: we mean that the algorithm is specified without any mention of the multicore parameters (p, M and B) and further, the PWS scheduler we use schedules tasks on available idle cores, without reference to the multicore parameters. Since multicores with a wide range of parameters are expected to appear on most desktops, such a resource oblivious feature in multicore algorithms appears to be helpful in supporting the portability of program codes. The PWS scheduler uses *work-stealing* [9, 7], where load balance is achieved by cores stealing tasks from other cores as needed.

Our main result, the SPMS sorting algorithm and its resource-oblivious performance, has the bounds stated below in Theorem 1. The algorithm proceeds in *rounds*, where a round, roughly speaking, corresponds to a parallel step. Our analysis uses the following parameters. We suppose that each core performs a single operation in $O(1)$ time, a cache miss takes at most b time, a steal request takes at most s time (whether successful or not), and the scheduler's work at the start of each round takes at most S time. We consider a multicore with p cores, each having a private cache of size M organized in blocks of size B, with all caches sharing an arbitrarily large global memory. The input, of size $n \geq Mp$ (this restriction ensures that both cores and caches can be fully utilized), is in the shared memory at the start of the computation, and SPMS is scheduled under PWS. Then:

Theorem 1. *On an input of length n, assuming $M \geq B^2$ (the 'tall cache'), for $p \leq \frac{n}{\max\{\log\log n,\ M\}}$, the sorting algorithm SPMS takes parallel time*

$$O\left(\frac{1}{p}\left(n\log n + b\cdot\frac{n\log n}{B\log M}\right) \ + \ (b+s+S)\log n\log\log n \ + \ b\beta(n,p,B)\right).$$

The fourth term, $\beta(n,p,B) = O(B\log n\log\log(n/p))$, is the block miss cost, and is bounded by the optimal sequential cache complexity provided $p \leq \frac{n}{B^2\log\log M\log M}$ (i.e., with a slightly 'taller' cache — $M \geq B^2\log B\log\log B$ suffices). This cost may also be reduced to match the optimal sequential cache complexity without this

additional restriction on p if system support is provided for the locking of a block during writes, and limiting the minimum task size to be at least B.

If we ignore the block miss cost for the moment, this bound represents the optimal work bound, plus the optimal cache miss bound, plus the critical path length times the cost of one cache miss plus one steal plus one scheduling event. Further, we note that there is no need for a global clock, or tight synchronization on the part of the cores, though the scheduler certainly imposes a significant degree of synchronization. The computation is entirely resource-oblivious in that the algorithm makes no mention of p, M or B, and PWS services idle cores without any reference to the number available or their cache parameters.

Extending the analysis of the randomized work stealer in [8, 1], we can obtain:

Theorem 2. *On an input of length n, assuming $M \geq B^2$, for $p \leq \frac{n}{\max\{\log\log n,\ M\}}$, the sorting algorithm SPMS when scheduled by the randomized work stealer, and taking into account both cache and block misses, takes expected parallel time*

$$O\left(\frac{1}{p}\left(n\log n + b\cdot\frac{n\log n}{B\log M}\right) \ + \ \frac{M}{B}\cdot\frac{b}{s}\left(bB\frac{\log n}{\log B} + (b+s)\log n\log\log n\right)\right).$$

(The analysis of this result can be found in [12, 14]).

Discussion. Our sorting algorithm is optimal in all respects except for the critical pathlength. The sorting algorithm for PEM in [4] achieves optimal $O(\log n)$ parallel steps, but is both cache- and core-aware. Achieving the same bound in a resource-oblivious manner appears considerably more challenging, and it is not clear if it is possible. We leave this as a topic for further research.

Another challenging topic is to extend our results to resource-oblivious scheduling on a multi-level caching hierarchy. Given the conflicting requirements of private and shared caches noted in [5, 10], it appears that some mechanism of supplying scheduler hints within the algorithm, and having a scheduler that uses the machine parameters effectively is needed. One such approach is used in [10]; however, that scheduler is *not* resource-oblivious, in contrast to our results.

In comparing our PWS scheduling to the PEM and Multi-BSP models, we note that these models both compute in a bulk-synchronous manner. We can easily adapt our results to work on either PEM or multi-BSP with the same performance as achieved with the PWS scheduler. However, our PWS framework adapts much more gracefully to differences in speeds among the cores than these bulk-synchronous models. Thus, if we have a few cores that execute faster than others (perhaps because they have smaller cache miss cost due to the cache layout), then PWS would enable the faster cores to take over (i.e, steal) work from the slower cores, balancing the work across the cores more effectively.

3 SPMS, A New Deterministic Sample, Partition, and Merge Sort

The heart of the algorithm is a procedure for computing a merging subproblem MS, whose input comprises r sorted lists $L_1, L_2, \cdots, L_r$, of total length m, with $m \leq r^c$, where $c \geq 6$ is a constant.

The sorting algorithm simply calls the merging procedure with $r = m = n$.

The merging algorithm performs two successive collections of recursive $\sqrt{r}$-way merges, each merge being on lists of total length at most $r^{c/2}$. To enable this, suitable samples of the input lists will be sorted by a logarithmic time procedure, which then allows the original problem to be partitioned into smaller subproblems that are merged recursively. More precisely:

Step 1. Partition the original problem MS into $k = O(m/r^{\frac{c}{2}-1})$ disjoint merging subproblems, $M_1, M_2, \cdots, M_k$, each comprising r sorted lists, with each subproblem having at most $r^{\frac{c}{2}}$ items in its r sorted lists. In addition, the items in M_i precede those in M_{i+1}, for $1 \leq i < k$.

Step 2. For each subproblem M_i, group its lists into disjoint subsets of $\sqrt{r}$ lists, and then in parallel merge the lists in each group. As M_i contains at most $r^{\frac{c}{2}}$ items, this bound applies to each of the individual groups too. Thus the $\sqrt{r}$-way merge in each group can be performed recursively. The output, for each subproblem M_i, is a collection of $\sqrt{r}$ sorted lists of total length at most $r^{\frac{c}{2}}$.

Step 3. For each subproblem M_i, recursively merge the $\sqrt{r}$ sorted lists computed in Step 2.

Step 1 details. The basic idea is to take a deterministic sample S of the input set comprising every $r^{\frac{c}{2}}$-th item in each list, to sort S, and to partition the r input lists about the items in S thereby forming smaller r-way merging subproblems. Some of these subproblems may have size as large as $r^{\frac{c}{2}+1}$, rather than the desired $r^{\frac{c}{2}}$. Any such subproblems are partitioned further, as needed, via samples S' of size $m'/r^{\frac{c}{2}-1}$ for each subproblem of size $m' \geq r^{\frac{c}{2}}$. The samples S and S' are sorted by performing all pairwise comparisons. More precisely:

Step 1.1. Let S comprise every $r^{\frac{c}{2}}$-th item in each of the input lists. Extract S from the input lists and then sort S, using a simple logarithmic time, quadratic work algorithm.

Step 1.2. Partition the r input lists $L_1, L_2, \cdots, L_r$ about S, creating subproblems $M'_1, M'_2, \cdots, M'_{k'}$, where $k' = |S| + 1$, and M'_i contains r sublists holding the items between the $(i-1)$th and ith items in S.

Step 1.3. Further partition any subproblem M'_i of size more than $r^{\frac{c}{2}}$, creating an overall collection of merging subproblems $M_1, M_2, \cdots, M_k$, each of size at most $r^{\frac{c}{2}}$, with the further property that the items in M_i precede those in M_{i+1}, for $1 \leq i < k$. This is done using a sample comprising every $r^{\frac{c}{2}-1}$-th item in M'_i.

Lemma 1. *The merging algorithm, on an input of r sorted lists of total length $m \leq r^c$, uses $O(m \log r)$ operations and $O(\log r \log\log r)$ parallel time, if $c \geq 6$.*

Proof. The parallel run time $T(r, m)$ is given by: $T(r, m) \leq \log r + 2T(\sqrt{r}, r^{c/2}) = O(\log r \log\log r)$.

Clearly, Steps 1.1 and 1.2 take $O(m)$ operations. To see the same bound applies to Step 1.3, we argue as follows. Each subproblem M'_i of size m' generates a sorting task of size $m'/r^{\frac{c}{2}-1} \leq r^{\frac{c}{2}+1}/r^{\frac{c}{2}-1} = r^2$. Performing all these sorting tasks requires at most $r^2 \cdot \sum m'/r^{\frac{c}{2}-1} \leq r^2 \cdot m/r^{\frac{c}{2}-1} \leq m$ operations, if $c \geq 6$.

Let $W(r, m)$ be the operation count for a collection of merging problems of total size m, where each comprises the merge of r lists of combined size at most r^c. Then we have: $W(r, m) \leq m + 2W(r^{1/2}, m) = O(m \log r)$.

Corollary 1. *The sorting algorithm, given an input of size n, performs $O(n \log n)$ operations and has parallel time complexity $O(\log n \log\log n)$, if $c \geq 6$.*

4 The Computation Model and PWS Scheduling

Before giving the resource oblivious implementation, we need to review the computation model and the PWS scheduling environment, mainly as developed in [12], although we make some changes here to address some new algorithmic features in SPMS.

The building blocks for our algorithms are computations on balanced binary trees such as for prefix sums. Such a computation is carried out by tasks: initially there is one task at the root of the tree; it forks two subtasks for each of its subtrees, and when they are done, it resumes and concludes the computation at the root. We will also use a tree of forking tasks to initiate a collection of parallel recursive calls, as needed in the merging and sorting algorithms.

Initially the root task for such a tree is given to a single core. Subtasks are acquired by other cores via task stealing. To this end, each core C has a task queue. It adds forked tasks to the bottom of the queue, while tasks are stolen from the top of the queue. So in particular, when C, on executing τ, generates forked tasks τ_1 and τ_2, it places the larger of τ_1 and τ_2 on its queue, τ_2 say, and continues with the execution of τ_1. This is a small generalization from [12], where the two forked tasks were assumed to be of the same size. When C completes τ_1, if τ_2 is still on its queue, it resumes the execution of τ_2, and otherwise there is nothing on its queue so it seeks to steal a new task. Except for one routine, our algorithm will be constructed from BP trees [12], which are trees of equal-sized forking nodes with an $O(1)$ operation computation at each node, and with the leaf nodes having either an $O(1)$ operation task or a recursive computation as their task. There is a mirror image tree for the joins which also performs $O(1)$ operations at each node. We will often talk of a subtree of the BP tree, when we really intend a subtree plus the mirror image subtree.

Let τ be a task associated with such a subtree. As in [12], by the size of τ, $|\tau|$, we mean the amount of data τ accesses in its computation. In contrast to [12], sometimes we will use the *virtual size* of τ, $\text{vs}(\tau)$; always $\text{vs}(\tau) \geq |\tau|$. Efficiency is ensured by the following BP tree property: if τ' is forked by τ, then $\text{vs}(\tau') \leq \frac{1}{2}\text{vs}(\tau)$.

To help with the scheduling, each node in a BP tree receives the integer priority $\log \text{vs}(\tau)$. These are strictly decreasing from parent to child. We will use the Priority Work-Stealing Scheduler (PWS) [12], which only allocates tasks of highest priority in response to steal requests. As noted in [12], task priorities are strictly decreasing on each task queue, and thus there will be at most one steal of a task of priority d from each core, and so at most p steals of tasks of priority d, for any d. This is key to bounding the overhead of the PWS scheduler.

As noted in Section 2, the I/O cost of an individual task τ is measured in *cache misses*, which we upper bound by how many blocks the core executing τ has to read into its cache, assuming none are present at the start of τ's execution, and *block misses*, which capture the cost of multiple cores writing to the same block.

As we shall see, each BP task τ in the sort algorithm incurs $O(\text{vs}(\tau)/B + \sqrt{\text{vs}(\tau)})$ cache misses when executed sequentially. Each task incurs only $O(B)$ block miss delay: for most of the tasks this follows from [12] because they engage in consecutive writes to a linear array; we will show that the remaining class of tasks will also incur only $O(B)$ block miss delay in their writing.

We will use the following bounds derived in [12] for a collection of parallel BP computations of total size n and sequential cache complexity Q, when scheduled under PWS. Here, the maximum (virtual) size of any root task in the BP collection is x, and any task of size s incurs $O(s/B + \sqrt{s})$ cache misses and shares $O(1)$ blocks with other tasks:

Fact 1. *For the I/O cost for a computation of the type stated above:*

1. *The cache miss bound is* $O(Q + p \cdot (\frac{\min\{M,x\}}{B} + \log \min\{x, B\} + \sqrt{x}))$.
2. *The block miss bound is* $O(p \cdot \min\{B, x\} \cdot (1 + \log \min\{x, \frac{n}{p}\})$.

The merging algorithm MS is built by combining (collections of parallel) BP computations, first by sequencing, and second by allowing the leaves of a BP tree to be recursive calls to the merging algorithm. This generalizes the above tree computation to a dag which is a series-parallel graph. The formal definition of such an 'HBP' computation is given in [12]. While we do not need the details of a general HBP computation here, we do need to define priorities carefully in view of the possible differences in the sizes of the recursive subproblems generated by a call to MS. We define these priorities in a natural way so that they are strictly decreasing along any path in the MS computation dag, and all tasks with the same priority have roughly the same size, as detailed in the full paper [13].

The recursive subproblems generated in Step 2 of MS need not be of the same size, so this portion of the algorithm does not exactly fit the BP frame-work of [12]. To handle this, we will first determine the cache-miss overhead for the natural parallel implementation of the algorithm, which we call the *ideal PWS costing*, and then add in the additional cache-miss cost for the PWS schedule. (The cost of block misses is discussed later.)

Definition 1. *The* ideal costing *assumes that a BP computation uses at most* $2n/M$ *cores, one for each distinct subtree of size* M *and one for each node ancestral to these subtrees.*

The cache miss cost in the ideal costing is $O(M/B)$ per subtree, plus $O(1)$ for each ancestral node, for a total of $O(n/B)$ cache misses.

We generalize this BP analysis to MS and SPMS by analyzing the algorithm in terms of parallel collection of tasks, each task of virtual size M. The cost of each task collection is bounded in turn: each task is costed as if it was allocated to a distinct core. As we will see, each such collection has total virtual size $O(n)$, and hence incurs $O((n/B) + \frac{n}{M}\sqrt{M})$ cache misses, which is $O(n/B)$ if $M \geq B^2$.

To analyze the cache miss cost of the PWS scheduling of MS, we separate the cost of steals of small tasks τ (those with $\text{vs}(\tau) \leq M$), which we bound later, and consider the impact of steals of large tasks. To bound this cost for a large stolen task τ, we overestimate by supposing that no small tasks are stolen from τ. Then (the possibly overestimated) τ executes one or more of the size M subtrees that are executed as distinct tasks in the ideal PWS costing, plus zero or more nodes ancestral to these subtree. Clearly, τ's cache miss cost is at most the sum of the cache miss costs of the corresponding distinct tasks in the ideal PWS costing. An analogous claim holds for the root task. Summing over the costs of the root task and of the large stolen tasks, yields that their total cost is bounded by the ideal PWS costing. This also bounds the processor-aware, cache-oblivious cost of MS, since the cost for block misses is minimal when there are no steals. We bound the cost of steals of small tasks using Fact 1, and results we derive here.

A final point concerns the management of local variables in recursive calls. We assume that if a task τ stolen by a core C has local variables, then the space allocated by the memory manager for these variables does not share any blocks with space allocated to other cores. Further, if the data resides in cache till the end of C's execution of τ, then the now unneeded local variables are not written back to the shared memory.

5 The Cache-Oblivious Parallel Implementation of SPMS

To achieve efficient oblivious performance, the merging algorithm MS needs to be implemented using tasks achieving the optimal ideal costing, as defined above. Many of the steps in MS are standard BP computations; their ideal costing is $O(n/B)$ and their PWS overhead can be bounded directly using Fact 1. However, here we address three types of computations in MS that do not fall within the framework in [12].

1. The recursion may well form very unequal sized subproblems. However, to achieve a small cache miss cost, the PWS scheduling requires forking into roughly equal sized subtasks. Accordingly we present the method of *grouping unequal sized tasks*, which groups subproblems in a task tree so as to achieve the balanced forking needed to obtain good cache-miss efficiency.

2. Balancing I/O for reads and writes in what amount to generalized transposes, which we call *transposing copies*. This issue arises in the partitioning in Steps 1.2 and 1.3. The challenge faced by a multicore implementation results from the delay due to the block misses, as discussed earlier.

3. One collection of tasks for sorting the samples in Step 1 uses non-contiguous writes. Fortunately, they use relatively few writes. We develop the *sparse writing* technique to cope.

Grouping Unequal Sized Tasks. We are given k ordered tasks $\tau_1, \tau_2, \cdots, \tau_k$, where each τ_i accesses $O(|\tau_i|/B)$ blocks in its computation (they are all recursive merges). Let $t_i \leq t_{ave}^2$ for all tasks, where t_{ave} is the average size of the tasks.

The tasks need to be grouped in a height $O(\log k)$ binary tree, called the *u-tree*, with leaves holding the tasks in their input order. The u-tree is used for

the forking needed to schedule the tasks. The u-tree will use virtual sizes for its scheduling subtasks and has the bounds given below. See [13] for more details.

Lemma 2. *The ideal PWS costing for scheduling the u-tree plus the cost of executing tasks τ_i of size M or less is $O(\sum_{i=1}^{k} t_i/B)$, where $t_i = |\tau_i|$.*

The Transposing Copy. The problem, given a vector A consisting of the sequence $A_{11}, \cdots, A_{1k}, \cdots, A_{h1}, \cdots A_{hk}$ of subvectors, is to output the transposed sequence $A_{11}, \cdots, A_{h1}, \cdots, A_{1k}, \cdots, A_{hk}$, where we are given that the average sequence length $l = |A|/hk \geq h$.

This is done by creating $\left\lceil \frac{|A_{ij}|}{l} \right\rceil$ tasks of virtual size l to carry out the copying of A_{ij}. The tasks are combined in column major order, i.e. in the order corresponding to destination locations. The full paper proves the following bound.

Lemma 3. *Let τ be a task copying lists of combined size s in the transposing copy. Then τ incurs $O(s/B + \sqrt{s})$ cache misses.*

Sparse Writing. Let A be an $s \times s$ array in which each of locations $c \cdot s$, $1 \leq c \leq s$ is written exactly once, but not in any particular order. A sequential execution incurs at most s cache misses.

Now consider a BP execution of this computation in which each leaf is responsible for one write. We claim that the I/O cost for all writes to A is $O(s^2/B + B)$ regardless of the ordering of the writes. We establish this bound as follows.

If $s \geq B$, each write into A incurs one cache miss, for a total cost of $O(s) \leq O(s^2/B)$ cache misses. There are no block misses in this case.

If $s < B$, there are only s accesses, but these can incur block misses. Let i be the integer satisfying $s \cdot i \leq B < s \cdot (i+1)$. Then, at most i writes occur within a single block. The worst-case configuration in terms of block miss cost occurs when there are i different sets of s/i writes, with each set having one write per block. Each such write to a block may incur a block wait cost equal to that of $\Theta(i)$ cache misses. Hence the overall delay in this case is at most that of $O(i^2 \cdot s/i) = O(s \cdot i) = O(B)$ cache misses.

5.1 Details of Step 1 in SPMS

Now, we describe parts of the algorithm in detail.

Each substep (except one) uses a BP computation or a collection BP computations running in parallel. We characterize the complexity of each size x (collection of) computations. Clearly it will have depth $O(\log x)$, and unless otherwise specified will incur $O(x/B)$ cache misses.

We begin with some notation. Let $L_1, L_2, \cdots, L_r$ be the r sorted input lists of total length $m \leq r^c$. The r lists are stored in sequential order. Let $S = \{e_1, e_2, \cdots, e_s\}$ comprise every $r^{\frac{c}{2}}$th item in the sequence of sorted lists; recall that S is sorted in Step 1.1, and then used to partition the r lists in Step 1.2.

Step 1.1. Sort S.

1.1.1. Construct arrays $S_1, S_2, \cdots, S_s$; each S_i is an array of length s which contains a copy of the sequence of elements in S.

(a) Compact the s samples within the list sequence $L_1, \cdots, L_r$, using prefix sums for the compaction. The result is an array $S_1[1..s]$ containing the s samples. This uses a sequence of 2 BP computations of size m.
(b) Form arrays S_i, $2 \leq i \leq s$, where each S_i is a copy of S_1. This is a BP computation of size $s^2 \leq m$.

1.1.2. In parallel for each i, compute rank of e_i in S_i.
First, for each S_i, compare e_i to each element in S_i. Then count the number of $e_j \geq e_i$, the desired rank of e_i in S_i. This uses two BP computations, and over all i, $1 \leq i \leq s$, they have combined size $O(s^2)$.

1.1.3. Create the sorted array $S[1..s]$ where $S[i]$ contains the element e_j with rank $\rho_j = i$.

The simple way to implement this step is for each element e_i to index itself into location $S[\rho_i]$. This will incur s cache misses, which can be argued is acceptable with a tall cache, since $s^2 = O(m)$. But this implementation could incur $s \cdot B$ block misses, which is excessive. To reduce the block miss cost, we split this step into two substeps:
(a) Initialize an all-zero auxiliary array $A'[1..m]$ and write each e_i into location $A'[\rho_i \cdot r^{c/2}]$.
This is the sparse writing setting analyzed earlier, and results in $O(s^2/B + B) = O(m/B + B)$ cache and block misses in a depth $\log s$ computation.
(b) Compact array A' into $S[1..s]$, which gives the desired sorted array of the samples. This is a prefix sums computation, a BP computation of size $O(m)$.

Step 1.2. Partition $L_1, L_2, \cdots, L_r$ about S. (Details in [13].)

Step 1.3. For each subproblem M'_i with $|M'_i| > r^{\frac{d}{2}}$ create a task to further partition M'_i. It is analogous to Steps 1.1 and 1.2 except that it uses a sample S' of size $m'_i/r^{\frac{d}{2}-1}$, where $m'_i = |M'_i|$.

Ideal PWS Costing. Summarizing the above discussion of the cache-miss costs for the merge (MS) gives the following bound for the number of cache misses in the ideal PWS costing.

Lemma 4. *In the ideal PWS costing, the merging algorithm MS, in performing a collection of merging tasks of total size $n \geq Mp$, in which each task comprises the merge of r lists of combined length at most r^c, incurs $O(\lceil \frac{n}{B} \rceil \left\lceil \frac{\log r}{\log M} \right\rceil)$ cache-misses, if $c \geq 6$ and $M \geq B^2$.*

Proof. As argued in the description of the algorithm, for each merging problem of size $m = \Omega(M)$, Substep 1 incurs $O(m/B + \sqrt{m}) = O(m/B)$ cache-misses, as $M \geq B^2$; smaller subproblems fit in cache and so incur $O(\lceil m/B \rceil)$ cache-misses.

Now let $C(r, n)$ be the cache-miss count for performing such a collection of merges for problems of merging r lists each of combined size at most r^c. Then, as the algorithm uses $O(n)$ space, we have, for a suitable constant $\gamma > 1$: for $n \leq \gamma M$: $C(r, n) = \lceil n/B \rceil$, and for $n \geq \gamma M$: $C(r, n) \leq \frac{n}{B} + 2C(r^{1/2}, n)$.

For a processor-aware, cache-oblivious implementation of SPMS, there is only a constant number of block misses per task executed by a core, costing $O(bB)$

per task, and the number of tasks in a p-core processor-aware implementation is $O(p \cdot \frac{\log n}{\log(n/p)})$. Thus, the block miss cost is dominated by the cache miss cost under our assumption that $n \geq Mp$ and $M \geq B^2$. Hence, with the above analysis and the parallel time bounds for the basic SPMS algorithm, as well as for the BP computations in the implementations given in this section, we obtain the result that SPMS can be scheduled on p cores, for $p \leq \frac{n}{\max\{M, \log\log n\}}$, to obtain optimal speed-up and cache-oblivious cache-efficiency. Note that in such a processor-aware schedule, there is no need for steals, and hence there is no further overhead beyond the cache-miss, block miss, and depth bounds that we have established for the computation.

We next establish that when scheduled under PWS, this implementation also achieves similar bounds resource-obliviously.

The Analysis of the PWS overhead. In addition to the results in Fact 1, the companion paper [12] shows that:

1. The cost of each up-pass is bounded by that of the corresponding downpass in BP and HBP algorithms.

2. The idle work (the time spent by a core when it is not computing nor writing on a cache or block miss), in a (parallel collection of) BP computations, aside the waiting already accounted for in the up-pass, is bounded by $O(p \cdot ((s + S + b) \log x + b \min\{x, B\}))$ where x is the size of the largest task in the collection, s bounds the time for a steal, S bounds the time to initiate a scheduling round, and b bounds the time for a cache miss.

3. Additional cache miss costs due to small tasks taking over the work of large tasks on an up-pass are bounded by the cache miss costs in the downpass.

And, as already noted in this paper, for the present algorithm:

4. The delay due to block misses for a stolen task τ is bounded by the time for $O(B)$ cache misses. This follows from results in Fact 1 for block misses, and from our method for Step 1.1.3, described earlier.

Lemma 5. *Let $M \geq B^2$. The delay $\mathrm{BM}_M(n, r^c)$ due to block misses in the merging algorithm for a collection of merging problems each of size at most r^c, and of total size $n \geq Mp$, is bounded by: $pB \log r^c (\log\log \frac{n}{p} - \log\log B)$ if $r^c \geq B$ and $B \leq \frac{n}{p} < r^c$, $pB \log r^c (\log\log \frac{r^c}{B} - \log\log B)$ if $r^c \geq B$ and $\frac{n}{p} \geq r^c$, and by $pr^c \log r^c$ if $n/p, r^c < B$.*

Proof. Using the bounds in Fact 1 for block misses, and since $M \geq B^2$ the top level BP computation causes the following number, $\mathrm{BMT}(n, r^c)$, of block misses: $pB \log \frac{n}{p}$ if $\frac{n}{p} \leq r^c$ and $r^c \geq B$, $pB \log r^c$ if $\frac{n}{p} > r^c$ and $r^c \geq B$, and $pr^c \log r^c$ if $r^c < B$.

Since $\mathrm{BM}_M(n, r^c) \leq \mathrm{BMT}(n, r^c) + 2\mathrm{BM}_M(n, r^{c/2})$, induction confirms the claimed bound.

Similar arguments bound the cache miss costs and idle time (see [13]). Adding these costs, plus those from Lemma 4 yields our main result.

Theorem 2. *When run on* $p \leq \min\{\frac{n}{\log\log n}, \frac{n}{M}\}$ *cores using the PWS scheduler on an input of size* $n \geq Mp$*, where* $M \geq B^2$*, the merging algorithm runs in time*

$$O(\frac{n\log n}{p} + \frac{bn}{Bp}\frac{\log n}{\log M} + \log n \log\log n(s + S + b) + bB\log n \log\log \frac{n}{p}).$$

The same bound applies to the sorting algorithm.

References

[1] Acar, U.A., Blelloch, G.E., Blumofe, R.D.: The data locality of work stealing. In: Theory of Computing Systems, vol. 35(3). Springer, Heidelberg (2002)

[2] Aggarwal, A., Vitter, J.S.: The input/output complexity of sorting and related problems. CACM 31, 1116–1127 (1988)

[3] Ajtai, M., Komlos, J., Szemeredi, E.: An O(n log n) sorting network. Combinatorica 3, 1–19 (1983)

[4] Arge, L., Goodrich, M.T., Nelson, M., Sitchinava, N.: Fundamental parallel algorithms for private-cache chip multiprocessors. In: ACM SPAA, pp. 197–206 (2008)

[5] Blelloch, G., Chowdhury, R., Gibbons, P., Ramachandran, V., Chen, S., Kozuch, M.: Provably good multicore cache performance for divide-and-conquer algorithms. In: ACM-SIAM SODA, pp. 501–510 (2008)

[6] Blelloch, G., Gibbons, P., Simhadri, H.: Brief announcement: Low depth cache-oblivious sorting. In: ACM SPAA. ACM, New York (2009)

[7] Blumofe, R., Leiserson, C.E.: Scheduling multithreaded computations by work stealing. JACM, 720–748 (1999)

[8] Blumofe, R.D., Leiserson, C.E.: Scheduling multithreaded computations by work stealing. Journal of the ACM 46(5), 720–748 (1999)

[9] Burton, F., Sleep, M.R.: Executing functional programs on a virtual tree of processors. In: ACM FPLCA, pp. 187–194 (1981)

[10] Chowdhury, R.A., Silvestri, F., Blakeley, B., Ramachandran, V.: Oblivious algorithms for multicores and network of processors. In: IEEE IPDPS (2010)

[11] Cole, R.: Parallel merge sort. SIAM J Comput 17(4) (1988)

[12] Cole, R., Ramachandran, V.: Efficient resource oblivious scheduling of multicore algorithms (2010) (Manuscript)

[13] Cole, R., Ramachandran, V.: Resource oblivious sorting on multicores. TR-10-13, Dept of Comp Sci, UT-Austin (2010)

[14] Cole, R., Ramachandran, V.: Resource oblivious sorting on multicores. (Submitted, 2010)

[15] Frigo, M., Leiserson, C.E., Prokop, H., Ramachandran, S.: Cache-oblivious algorithms. In: IEEE FOCS (1999)

[16] Valiant, L.G.: A bridging model for multi-core computing. In: Halperin, D., Mehlhorn, K. (eds.) Esa 2008. LNCS, vol. 5193, pp. 13–28. Springer, Heidelberg (2008)

Interval Sorting*,**

Rosa M. Jiménez and Conrado Martínez

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Barcelona, Spain
{jimenez,conrado}@lsi.upc.edu

Abstract. In the interval sort problem, we are given an array A of n items and p ranges of ranks $I_1 = [\ell_1, u_1]$, ..., $I_p = [\ell_p, u_p]$. The goal is to rearrange the array so that $A[\ell_t..u_t]$ contains the ℓ_t-th, ..., u_t-th smallest elements of A in nondecreasing order, for all t, $1 \le t \le p$, and $A[u_t + 1..\ell_{t+1} - 1]$ contains the $(u_t + 1)$-th, ..., $(\ell_{t+1} - 1)$-th smallest elements of A, for all t, $0 \le t \le p$. That is, the array is sorted by blocks, with sorted and unsorted blocks alternating. One of the most interesting aspects of this research is the unification of several important and related problems (sorting, selection, multiple selection, partial sorting) under a single framework. Results on interval sorting generalize the results for any of these particular—and fundamental—problems.

We propose a divide-and-conquer algorithm, owing to quicksort and quickselect, named *chunksort*, to solve the problem. We give an exact expression for the average number of comparisons made by the basic variant of chunksort. Then we consider what is the expected optimal number of comparisons needed to solve an interval sort instance and we design a variant of chunksort that achieves near optimal expected performance, up to $n + o(n)$ comparisons. In fact, we conjecture that the algorithm that we propose has actually optimal expected performance up to $o(n)$ terms and provide some evidence for this conjecture.

1 Introduction

We call *interval sorting* the following problem: given an array A of n items drawn from a totally ordered domain, and a set $I = \{[\ell_t, u_t] \,|\, 1 \le t \le p\}$ of p disjoint intervals, with

$$1 \le \ell_1 \le u_1 < \ell_2 \le u_2 < \cdots < \ell_p \le u_p \le n,$$

the goal is to rearrange the array A in such a way that $A[\ell_t..u_t]$ contains the ℓ_t-th, $(\ell_t + 1)$-th, ..., u_t-th smallest elements of A in ascending order, for all t,

* This research was supported by the Spanish Min. of Science and Technology project TIN2007–66523 (FORMALISM) and the Catalonian Government program for research groups under contract 2009 SGR 1137.

** We thank Amalia Duch for her useful comments and suggestions.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 238–249, 2010.

$1 \le t \le p$, and furthermore, $A[u_t + 1..\ell_{t+1} - 1]$ contains the $(u_t + 1)$-th, $(u_t + 2)$-th, ..., $(\ell_{t+1} - 1)$-th smallest elements of A, but not in any particular order, for all t, $0 \le t \le p$ (we use the convention $u_0 = 0$ and $\ell_{p+1} = n + 1$).

In other words, if we define $I_t = [\ell_t, u_t]$, $\overline{I}_t = [u_t + 1, \ell_{t+1} - 1]$, and if $A[R]$ denotes the subarray with indices in the range given by R, after interval sorting, we have

$$A[\overline{I}_0] \le A[I_1] \le A[\overline{I}_1] \le A[I_2] \le \cdots \le A[I_p] \le A[\overline{I}_p],$$

where $A[R] \le A[S]$ means that for any two items $x \in A[R]$ and $y \in A[S]$, we have $x \le y$. Additionally, each $A[I_t]$ is sorted in nondecreasing order.

Along the paper we will find convenient to talk about *gaps*, the $\overline{I}_t$'s, as opposed to the intervals, the I_t's. A subarray $A[I_t]$ or $A[\overline{I}_t]$ will be sometimes called a *block*. We will denote $m_t = u_t - \ell_t + 1$ the size of the t-th interval and $\overline{m}_t$ the size of the t-th gap. We shall also use $m = m_1 + \cdots + m_p$ for the total number of items to be sorted and $\overline{m} = \overline{m}_0 + \cdots + \overline{m}_p = n - m$ for the total number of elements that will remain "unsorted".

The reader will have undoubtedly noticed that interval sorting generalizes both sorting, with $p = 1$, $I_1 = [1, n]$, and selection of the j-th, with $p = 1$, $I_1 = [j, j]$. It also generalizes the problem of multiple selection, setting $I_1 = [j_1, j_1], \ldots, I_p = [j_p, j_p]$, and the problem of partial sorting, setting $p = 1$ and $I_1 = [1, m]$ for some $m \le n$.

Sorting the whole array with worst-case time in $\Theta(n \log n)$ is an obvious solution for the interval sorting problem, but in many cases, when $m \ll n$ it will be wasteful. Here, we consider better alternatives.

In this paper, we propose the algorithm *chunksort* to solve the interval sorting problem. Chunksort is a simple divide-and-conquer algorithm in the spirit of the well-known quicksort and quickselect by Hoare [2,3]. It "adapts" its behavior to the set of intervals provided, performing exactly as quicksort, quickselect, multiple quickselect [10] or partial quicksort [6] when appropriate. In Section 2 we review the basic variant of the chunksort algorithm. Then, in Section 3 we consider its expected performance. Section 4 addresses the question of optimally solving an interval sort problem. We first consider a simple lower bound on the complexity of comparison-based algorithms for interval sorting. Then we propose a variant of chunksort that achieves almost expected optimal performance (up to $n + o(n)$ comparisons). We finally discuss in Section 5 our conjecture that the algorithm of Section 4 actually achieves optimal expected performance (up to $o(n)$ comparisons).

2 Chunksort

The input to the `chunksort` algorithm is a pair of arrays $A[1..n]$, $I[1..p]$ and indices i, j, r and s, which delimite the corresponding subarrays $A[i..j]$ and $I[r..s]$. Each component of I is a pair encoding the lower and upper ends of an interval, thus $I[t].\ell = \ell_t$ and $I[t].u = u_t$. Moreover, we assume that we store two sentinels $I[0].u = 0$ and $I[p + 1].\ell = n + 1$.

Algorithm 1. Chunksort

```
procedure CHUNKSORT(A, i, j, I, r, s)
    if i ≥ j then return        ▷ A contains one or no elements
    end if
    if r ≤ s then
        pv ← SELECTPIVOT(A, i, j)
        PARTITION(A, pv, i, j, k)
        t ← LOCATE(I, r, s, k)
     ▷ Locate the value t such that ℓ_t ≤ k ≤ u_t with I_t = [ℓ_t, u_t],
     ▷ or u_t < k < ℓ_{t+1}
        if u_t < k then        ▷ k falls in the t-th gap
            CHUNKSORT(A, i, k − 1, I, r, t)
            CHUNKSORT(A, k + 1, j, I, t + 1, s)
        else      ▷ k falls in the t-th interval
            CHUNKSORT(A, i, k − 1, I, r, t)
            CHUNKSORT(A, k + 1, j, I, t, s)
        end if
    end if
end procedure
```

At some recursive stage, the algorithm considers some subarray $A[i..j]$, where it has to sort the blocks defined by the intervals $I[r..s]$. The algorithm picks an element from the subarray $A[i..j]$, called the *pivot*, and rearranges the subarray so that the pivot lands at position k, $i \le k \le j$. All elements in $A[i..k-1]$ are smaller than or equal to $A[k]$, and all elements in $A[k+1..j]$ are greather than or equal to $A[k]$. The next step is to locate the interval or the gap where the pivot has fallen. The procedure `locate` returns t, with $\ell_t \le k \le u_t$ or $u_t < k < \ell_{t+1}$.

If the pivot is in a gap ($u_t < k < \ell_{t+1}$), we have to call recursively the procedure on the left subarray $A[i..k-1]$ with all intervals to the left of u_t, including $I_t = [\ell_t, u_t]$, and we have also to call the procedure on the right subarray $A[k+1..j]$ with all intervals from $I_{t+1} = [\ell_{t+1}, u_{t+1}]$ onwards. If the pivot has fallen inside an interval then we do about the same, but the interval $I_t = [\ell_t, u_t]$ participates in the two recursive calls. Actually, we might "tune" the algorithm replacing the interval $I_t = [\ell_t, u_t]$ by two intervals $[\ell_t, k-1]$ and $[k+1, u_t]$, using them for the recursive calls on $A[i..k-1]$ and $A[k+1..j]$, respectively. It is not difficult to see that the algorithm already behaves just as if such replacement had taken place, since any position which is outside the bounds i and j is simply ignored.

We can apply several standard techniques to improve `chunksort`, including recursion removal, recursion cutoff on small subfiles (the easiest thing to do would be to fully sort small subfiles using insertion sort), sampling for pivot selection, loop unwrapping, reordering the recursive calls to minimize stack space, etc. Also, in a finely tuned implementation of `chunksort` the location of the cutting point t within I should be done using binary search for peak performance.

In order to achieve near optimal performance with chunksort, we will see in Section 4 that the algorithm must pick pivots which land exactly or very near

to interval ends. That is, in each recursive stage, the final position k of the pivot should be equal to ℓ_t or u_t, for some t, or very close.

Then it makes sense to use the following technique. If $k = \ell_t$ for some t, then instead of the calls $\textsc{Chunksort}(A, i, k-1, I, r, t)$ and $\textsc{Chunksort}(A, k+1, j, I, t, s)$, we can make the call to chunksort in the left subarray with the intervals I_r to I_{t-1} since no element of I_t participates in this subproblem, and we can also change the lower bound ℓ_t to $\ell_t + 1$. Similarly, if $k = u_t$, then we can call to chunksort in the right subarray with the intervals I_{t+1} to I_s, and change the upper bound u_t to $u_t - 1$. Furthermore, if the t-th interval contains a single element and we had $k = \ell_t = u_t$ then we have "thrown" this interval away. Notice that although the optimization works for any variant of chunksort, in practical terms it does not pay off to include it except when pivots land often at an extreme of some interval. For instance, this is not the case with the basic variant which chooses pivots at random.

3 The Average Performance of Chunksort

In this section we will derive the general recurrence for the average number of comparisons/exchanges/passes made by chunksort. Let $C_{n;\{I_r,\ldots,I_s\}}$ be the average number of comparisons/exchanges/passes/... made by chunksort on an array of size n and the intervals $I_r, \ldots, I_s$. We use also $C_n = C_{n;\{I_1,\ldots,I_p\}}$. We assume that all elements are distinct and, as it is usual in the analysis of comparison-based sorting and selection algorithms, that all $n!$ possible arrangements of the n elements are equally likely.

When the array is empty we have nothing to do, so $C_{0;\{I_r,\ldots,I_s\}} = 0$. Likewise, if $r > s$ then we assume $C_{n;\{I_r,\ldots,I_s\}} = 0$. Assume now that $n > 0$ and $r \le s$. Then the recurrence is almost self-explanatory. We consider all possible cases for the value of t and whether the pivot landed at I_t or at $\overline{I}_t$; this yields

$$C_{n;\{I_r,\ldots,I_s\}} = T_n + \sum_{t=r-1}^{s} \sum_{k \in \overline{I}_t} \pi_{n,k} \big(C_{k-1;\{I_r,\ldots,I_t\}} + C_{n-k;\{I_{t+1},\ldots,I_s\}} \big) + \sum_{t=r}^{s} \sum_{k \in I_t} \pi_{n,k} \big(C_{k-1;\{I_r,\ldots,I_t\}} + C_{n-k;\{I_t,\ldots,I_s\}} \big), \quad (1)$$

where $\pi_{n,k}$ is the probability that the selected pivot is the k-th of the n elements. Different pivot selection strategies will yield particular *splitting probabilities* $\pi_{n,k}$. On the other hand, in a more general setting, the recurrence will be valid for $n \ge n_0$, if we switch to a different and simpler algorithm when the input is small enough, namely, when $n < n_0$. Last but not least, the toll function T_n will depend on the quantity of interest, be it comparisons, exchanges, recursive passes, etc. Notice that setting $r = s = 1$ —that is, only one interval— and the values ℓ_1 and u_1, we can easily specialize the recurrence above for quicksort, quickselect and partial quicksort.

Here we should only consider the standard variant, hence $n_0 = 1$ and $\pi_{n,k} = 1/n$. That is, since the pivot is chosen at random (or equivalently, we assume

that the input array A contains a random permutation), the probability that it is the k-th out of n is the same for all values of k. On the other hand, we will only consider the comparisons between elements made while partitioning the array A around the pivot, i. e., we will set $T_n = n - 1$. We are disregarding thus all the comparisons between indices, and in particular, those necessary to locate the interval or gap containing the position of the pivot after partitioning. It might be argued that since this number is $o(n)$ —either because p is very small, e. g., constant, or because we are using binary search— the main order terms in the average performance of chunksort are not affected.

The solution of (1) is by no means trivial. In particular, we made an educated guess from the known results for partial quicksort [6] and multiple quickselect [10], which then we checked by induction. An alternative path is to solve the cases for one interval, two intervals, etc. (see [5]). A pattern among the solutions emerges, and the general form for the solution can be hypothesized and then checked by induction. If we follow that path we would first introduce a generating function

$$C(z; x, y) = \sum_{n \geq 0} \sum_{1 \leq i \leq j \leq n} C_{n;\{[i,j]\}} z^n x^i y^j,$$

to obtain the average performance when there is a single interval $I_1 = [i, j]$. Recurrence (1) translates to a linear differential equation similar to those satisfied by quickselect and partial quicksort, and can be solved using similar techniques. The crucial point to observe here is that, if $i = 1$ or $j = n$ this problem reduces to that of partial quicksort, whose solution can be found in [6]. Here one obtains (see also [5])

$$C_{n;\{[i,j]\}} = 2n + 2(n+1)H_n - 6(j-i) + 6 - 2(i+2)H_i - 2(n+3-j)H_{n+1-j}.$$

Going to the next case, $p = 2$, the formula is very similar, but the size of the intermediate gap $\overline{m}_1 = \ell_2 - u_1 - 1$ gets involved. We omit here the tedious and laborious details and provide the general solution, which is given in the following theorem.

Theorem 1. *The average number of element comparisons made by chunksort to arrange an array of size n, given p intervals $I_t = [\ell_t, u_t]$, $1 \leq t \leq p$ is*

$$C_n = 2n + u_p - \ell_1 + 2(n+1)H_n - 7m - 2 + 15p$$
$$- 2(\ell_1 + 2)H_{\ell_1} - 2(n + 3 - u_p)H_{n+1-u_p} - 2\sum_{t=1}^{p-1}(\overline{m}_t + 5)H_{\overline{m}_t+2},$$

where $\overline{m}_t = \ell_{t+1} - u_t - 1$, is the size of the t-th gap $\overline{I}_t = [u_t + 1, \ell_{t+1} - 1]$, $0 \leq t \leq p$, and $m = m_1 + m_2 + \cdots + m_p$ is the combined size of the blocks which are internally sorted, i. e., $m_t = u_t - \ell_t + 1$ is the size of the t-th interval $I_t = [\ell_t, u_t]$, $1 \leq t \leq p$.

The reader can readily convince herself that the formula works for quicksort ($p = 1$, $\ell_1 = 1$, $u_1 = n$), for quickselect ($p = 1$, $\ell_1 = u_1 = j$), for partial quicksort ($p = 1$, $\ell_1 = 1$, $u_1 = m$) and multiple quickselect ($\ell_t = u_t = j_t$, for $1 \leq t \leq p$).

4 Optimal Interval Sorting

We start first with a simple lower bound on the minimum expected number $\Lambda(n, \mathbf{m}, \overline{\mathbf{m}})$ of comparisons needed to solve interval sorting using any comparison-based algorithm. Here, $\mathbf{m} = (m_1, \ldots, m_p)$ and $\overline{\mathbf{m}} = (\overline{m}_0, \ldots, \overline{m}_p)$ are the two vectors and the value n univocally determining the interval sorting instance.

To begin with, if we apply interval sort to the array and then we optimally sort the items that fall in the gaps we obtain a sorted array, but the total number of comparisons must be at least the minimum number of comparisons necessary to sort the array at once:

$$\Lambda(n, \mathbf{m}, \overline{\mathbf{m}}) + \sum_{t=0}^{p} \log_2(\overline{m}_t!) \geq \log_2(n!).$$

Then,

$$\begin{aligned}
\Lambda(n, \mathbf{m}, \overline{\mathbf{m}}) &\geq n \log_2 n - \sum_{t=0}^{p} \overline{m}_t \log_2 \overline{m}_t - m \log_2 e + o(n) \\
&= \sum_{t=1}^{p} m_t \log_2 n + \sum_{t=0}^{p} \overline{m}_t \log_2 n - \sum_{t=0}^{p} \overline{m}_t \log_2 \overline{m}_t - m \log_2 e + o(n) \\
&= \sum_{t=1}^{p} m_t \left(\log_2 m_t - \log_2 \frac{m_t}{n} \right) - \sum_{t=0}^{p} \overline{m}_t \log_2 \frac{\overline{m}_t}{n} - m \log_2 e + o(n) \\
&= \sum_{t=1}^{p} m_t \log_2 m_t + n\mathcal{H}\left(\{\overline{m}_0/n, m_1/n, \overline{m}_1/n, \ldots, m_p/n, \overline{m}_p/n\}\right) \\
&\quad - m \log_2 e + o(n)
\end{aligned}$$

with $\mathcal{H}(\{q_t\}) = -\sum_t q_t \log_2 q_t$ denoting the entropy of the discrete probability distribution $\{q_t\}$; note that the q_t's must sum 1.

Now, suppose that we had an algorithm which given an array of n elements and a rank j, partitioned the array around an element of rank j' very close to j, that is, $|j - j'| = O(\sqrt{n})$, using $n + o(n)$ comparisons with very high probability.

With this algorithm as a building block, a solution to the interval sort problem would consist of two phases. In the first phase, $2p$ pivots are brought to positions $\ell_t - \delta_t$ and $u_t + \epsilon_t$, for all t, $1 \leq t \leq p$, for some "small" δ_t and ϵ_t, say, $\delta_t + \epsilon_t = O(\sqrt{n})$. Thus the original array is rearranged in such a way that

$$A[\overline{I}'_0] \leq A[I'_1] \leq A[\overline{I}'_1] \leq \ldots A[I_p] \leq A[\overline{I}'_p],$$

with $I'_t = [\ell_t - \delta_t .. u_t + \epsilon_t] \supseteq I_t$, $1 \leq t \leq p$, and $\overline{I}'_t = [u_t + \epsilon_t + 1 .. \ell_{t+1} - \delta_{t+1} - 1] \subseteq \overline{I}_t$, $0 \leq t \leq p$.

In the second phase the elements in the ranges I'_t must be optimally sorted; since the size of I'_t differs by only a small amount from the size of I_t, the cost of the second phase is $\sum_{t=1}^{p} m_t \log_2 m_t + O(p\sqrt{n})$.

Both phases can be carried out by the chunksort algorithm, provided that we carefully redesign the selection of pivots at each recursive stage. For example, when the number of elements to be sorted by some recursive call is not much smaller than the size of the corresponding subarray, the pivot should be close to the median of the subarray.

Consider the call CHUNKSORT(A, i, j, I, r, s). The size of the subarray is $N = j - i + 1$ and the number of elements to be sorted is

$$M = m_{r+1} + \ldots + m_{s-1} + u_r - \max(i, \ell_r) + 1 + \min(u_s, j) - \ell_s + 1.$$

If $N - M = o(N/\log N)$ then it is more efficient to sort the whole subarray. In that case, we would select a sample of $\Theta(\sqrt{N})$ elements and choose the median of the sample as the pivot. After partitioning, the subsequent calls to chunksort will behave in a similar way, because they receive subarrays where the number of items to be sorted on each is close to their corresponding sizes. In this way, we will sort the subarray $A[i..j]$ with an optimal expected number of comparisons $N \log_2 N + o(N \log N)$ [8].

Because of a similar argument, we assume that all gaps have size $\Omega(n/\log n)$; if not, we will do better by discarding small gaps separating two intervals, i.e., merging the two intervals into a single one. The elements of the discarded gaps will be sorted even though that was not required. But the time to put pivots to delimit those gaps would be greater than the time needed to sort their elements.

How do we have to select the pivots when $M \ll N$? If we want to put a pivot close to some rank ρ, we should use a similar technique to that in the previous case: take a random sample of $s = \Theta(\sqrt{N})$ elements, and choose the element in the sample whose rank within the sample is $\approx \frac{\rho}{N} \cdot s$. Actually, if $\rho = \ell_t$ for some t, we will take the element in the sample whose rank is $\frac{\rho}{N} \cdot s - \Delta$, for some positive $\Delta = o(s)$, to guarantee with high probability that the choosen pivot has rank $\ell_t - \delta_t$ for $\delta_t = O(\sqrt{N})$. Likewise, if $\rho = u_t$ for some t, we can take the element in the sample whose rank is $\frac{\rho}{N} \cdot s + \Delta$, to guarantee with high probability that the choosen pivot has rank $u_t + \epsilon_t$ with $\epsilon_t = O(\sqrt{N})$ [7].

It remains thus the problem of deciding which is the rank ρ that we should choose at each recursive step. For that purpose, the "iron bar cutting" metaphor turns out to be very useful. Thanks to sampling we can break the original array ("the iron bar") at designated marks ("the interval endpoints"). In this sense, our problem is to cut the iron bar into pieces, making a cut at each given mark. The resulting pieces will then be either left "unpainted" (= unsorted) or they will be "painted" (= sorted). Once the bar is cut at the extremes of some piece to be painted, our procedure takes care of sorting it optimally, so that now we can concentrate on the optimal way to produce the cuts, since the order in which we cut the bar matters.

Let $\rho_r, \ldots, \rho_s$ be the set of endpoints we are dealing with in a given moment: ρ_r and ρ_s delimite the subarray and the cuts must be done at $\rho_{r+1}, \ldots, \rho_{s-1}$. We take $\rho_{2t} = u_t$ and $\rho_{2t+1} = \ell_t$, for $0 \le t \le p$. Let $c(r, s)$ denote the cost of cutting the subarray at $\rho_{r+1}, \rho_{r+2}, \ldots, \rho_{s-1}$ optimally. We are interested in $c(0, 2p+1)$, with $\rho_0 = 0$ and $\rho_{2p+1} = n + 1$.

If $r+1 \geq s$ there is nothing to cut, so $c(r,s)=0$. If $r+1<s$ then there is 1 or more endpoints and the optimal cost is given by

$$c(r,s) = \rho_s - \rho_r + \min_{r<t<s}(c(r,t)+c(t,s)).$$

Here we take the size of the block delimited by the endpoints (not including them), plus one, as the cost of partitioning the block. The cost should also include the cost of sampling to select the pivot, but since it is asymptotically smaller than the cost of partitioning we can safely ignore it, as this will not affect the main order term of that cost. Finding the optimal costs $c(r,s)$ and hence the optimal choices for ρ_t at each recursive stage can be easily accomplished using dynamic programming. The space requirements are $\Theta(p^2)$ and the time for the computation is $\Theta(p^3)$. Let $\omega(r,s)=\rho_s-\rho_r$. Since the "weights" $\omega(r,s)$ satisfy the quadrangle inequality and $\omega(r,s)\leq\omega(r',s')$ whenever $[r,s]\subseteq[r',s']$, it follows that Knuth-Yao's technique [11] can be applied and then the optimal costs and optimal cutting points can be computed in $\Theta(p^2)$. Let $\rho(r,s)$ denote the optimal choice to minimize $c(r,s)$ when $r+1<s$. Dynamic programming gives us these optimal choices, so we can carry out a preprocess to compute the values $\rho(r,s)$; chunksort will only need to access the array that stores those values to guide its choices.

In what follows we will show that

$$c(0,2p+1) \leq (n+1)\mathcal{H}\left(\{\overline{m}_0/n, m_1/n, \overline{m}_1/n, \ldots, m_p/n, \overline{m}_p/n\}\right) + O(\log n). \quad (2)$$

This implies that the variant of chunksort that we have explained here has almost optimal expected performance up to n plus lower order terms and provided that $p=o(\sqrt{n})$. Indeed, with $H := \mathcal{H}\left(\{\overline{m}_0/n, m_1/n, \overline{m}_1/n, \ldots, m_p/n, \overline{m}_p/n\}\right)$, we have

$$\begin{aligned}\sum_{t=1}^{p} m_t \log_2 m_t + n\cdot H - m\log_2 e + o(n) &\leq \Lambda(n,\mathbf{m},\overline{\mathbf{m}}) \\ &\leq \sum_{t=1}^{p} m_t \log_2 m_t + c(0,2p+1) + O(p\sqrt{n}) \\ &\leq \sum_{t=1}^{p} m_t \log_2 m_t + n\cdot H + n + \text{lower order terms.}\end{aligned}$$

In order to prove (2) we will consider the recurrence for the cost of the first phase of chunksort in more general terms:

$$\hat{c}(r,s) = \rho_s - \rho_r + \hat{c}(r,t) + \hat{c}(t,s), \qquad r+1<s$$

where ρ_t is the endpoint choosen at the corresponding recursive stage; we do not insist now that ρ_t is the optimal cutting point. If we have a look at the recursion tree for chunksort, then the root node has weight $\omega(0,2p+1)$ and it is labeled with the index t of the choosen endpoint. Its left subtree is the tree

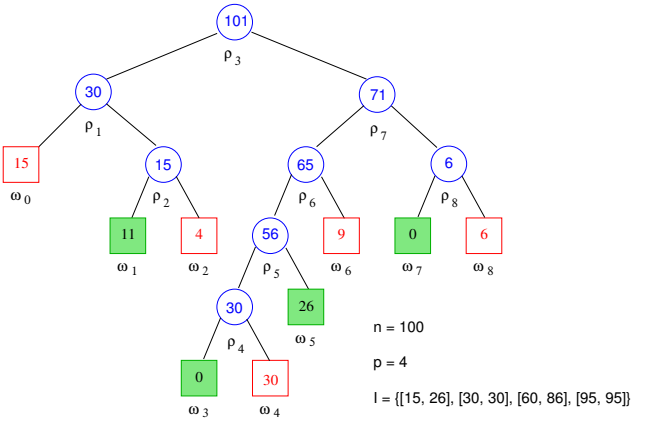


Fig. 1. A recursion tree for chunksort and the associated cost

that corresponds to the partition of the subarray delimited by ρ_0 and ρ_t, while the right subtree corresponds to the partition of the subarray delimited by ρ_t and ρ_{2p+1}, and so on. When the array to partition is delimited by ρ_t and ρ_{t+1} it contains no endpoints to partition around, and the corresponding subtree is a leaf contributing zero to the cost. If t is even, then ρ_t and ρ_{t+1} delimit a gap and chunksort will not do anything else there. When t is odd, ρ_t and ρ_{t+1} are the boundaries of an interval and chunksort will sort (optimally) that interval; however, we count that cost of sorting intervals separately.

In summary, the sum of the weights in the internal nodes give us the cost of that particular way of partitioning until gaps and intervals have been "isolated". Of course, we must have that this cost is equal to or greater than $c(0, 2p + 1)$. Let $\omega_t = \omega(t, t + 1)$ for $0 \leq t \leq 2p$. Given such a tree, denote d_t the depth of the t-th leaf, $0 \leq t \leq 2p$. Then the cost of the tree (see Fig. 1) is

$$C = \sum_{v \text{ is an internal node}} \omega(v) = \sum_{t=0}^{2p} d_t \omega_t$$

And the cost $c(r, s)$ is the cost of the optimal tree C_{opt}, that is, the one that minimizes the quantity above. In the example of Fig. 1, internal nodes and leaves are labelled by its corresponding $\omega(r, s)$. Shaded leaves correspond to intervals; blank leaves correspond to gaps. The endpoint choosen for partition at each recursive call is depicted next to is corresponding internal node.

If we define $\beta_t = \omega_t/(n + 1)$ then $C = (n + 1)P$, with $P = \sum_{t=0}^{2p} d_t \beta_t$; that is, P is the external weighted path length.

There are many results on the construction of optimal binary search trees and near-optimal binary search trees (see, for instance [9] and the references therein). Here, we face the particular situation where we have to construct an optimal or near-optimal binary search tree to access its leaves with probabilities

of access $\beta_0, \beta_1, \ldots, \beta_{2p}$; the probabilities of access to internal nodes are in our case all zero.

There are two important reasons to consider now heuristics to construct near optimal trees. On the one hand, they provide a simple and more efficient alternative to the dynamic programming to choose a good —if not optimal— endpoint at each recursive stage of chunksort. On the other hand, we have upper bounds on the external weighted path length of the tree constructed by such heuristics, namely [9]

$$P_{\text{heur}} \leq H' + 1, \qquad H' = \mathcal{H}(\{\beta_t\}).$$

This entails the sequence of inequalities

$$C_{\text{opt}} \leq C_{\text{heur}} \leq (n+1)(H'+1) \leq n \cdot H + n + O(\log n),$$

since $H' = H + O\left((\log n)/n\right)$. Indeed, we have

$$\beta_{2t} = \frac{\overline{m}_t + 1}{n+1}, \quad 0 \leq t \leq p; \qquad \beta_{2t-1} = \frac{m_t - 1}{n+1}, \quad 1 \leq t \leq p,$$

and manipulating the expression for H', removing the terms for which $m_t = 1$, and using easy bounds on $\log(1+x)$ and $\log(1-x)$, yields $H' = H + O((\log n)/n)$.

To conclude this section, we briefly discuss a couple of heuristic algorithms that can be used to guide chunksort's choice of endpoints at each recursive stage; using any of them will only incur an additional cost of $O(p \log p)$, while the overall expected cost of the resulting chunksort is still near optimal. Both heuristics are simple adaptations of the heuristics that have been proposed for the construction of near optimal binary search trees [9].

In one of them, we look for the endpoint ρ_t, $r < t < s$, closest to the middle of the range $i..j$. If more than one endpoint satisfies this criterion and all them are $\leq \lceil (j-i+1)/2 \rceil$ then we take the one with largest index; if all the closest endpoints are $> \lceil (j-i+1)/2 \rceil$ then we take the one with smallest index. If $\rho_t \leq \lceil (j-i+1)/2 \rceil \leq \rho_{t+1}$ and both endpoints are at the same distance of the middle of the range, then we take any of them, say the one with smallest index. This heuristic is easy to implement; it introduces an additional cost $O(\log p)$ at each recursive stage, for a total cost $O(p \log p)$.

The other heuristic is slightly more complicated, as each recursive call needs to know its level ℓ (the first call is at level 0), and a reference point, initially $\lceil n/2 \rceil$. In a recursive call at level ℓ, we look for the endpoint closest to the reference point r, and use the same criteria as before to decide ties. Once the endpoint ρ_t is choosen and the partition around the pivot has been carried out, the subsequent two recursive calls are made with reference points $r - \lfloor n/2^{\ell+1} \rfloor$ and $r + \lceil n/2^{\ell+1} \rceil$ and level $\ell + 1$. This heuristic can also be implemented with cost $O(\log p)$ to locate a good endpoint for each recursive stage.

5 Final Remarks

We conjecture that the variant of chunksort described in Section 4 has actually expected optimal performance, up to lower order terms, more specifically if the

dynamic programming algorithm is used to choose cutting points. Indeed, neither the lower bound on $\Lambda(n, \mathbf{m}, \overline{\mathbf{m}})$ nor the upper bound on the cost of the optimized chunksort are tight. It is known [9] that the difference $P_{\text{heur}} - P_{\text{opt}} \leq \log_2 P_{\text{opt}} \approx \log_2 H$. Suppose that P_{heur} and P_{opt} differ by some small constant c. Then C_{heur} and C_{opt} would differ by $c(n+1)$, and the gap between the lower bound and the cost of optimized chunksort would shrink accordingly.

Additional arguments supporting this conjecture come from considering particular cases, namely, selecting the j-th smallest element out of n. This is the particular case of interval sort with $p = 1$ interval $I_1 = [j, j]$. Therefore, $\overline{m}_0 = j - 1$ and $\overline{m}_1 = n - j$. It is well known [1] that the expected number of comparisons needed is at least $n + \min(j-1, n-j) +$ lower order terms, while our lower bound gives

$$n\left(-\frac{j-1}{n}\log_2\frac{j-1}{n} - \frac{n-j}{n}\log_2\frac{n-j}{n}\right) + \text{lower order terms}$$

so the difference is at least $(2 - \ln 3/\ln 2)n + o(n) \approx 0.415n + o(n)$, for any j. On the other hand,

$$\begin{aligned} c(0,3) &= \rho_3 - \rho_0 + \min_{0<t<3}(c(0,t) + c(t,3)) \\ &= n + 1 + \min(c(0,1) + c(1,3), c(0,2) + c(2,3)) \end{aligned}$$

Since $c(0,1) = c(2,3) = 0$, $c(1,3) = n + 1 - j$ and $c(0,2) = j$, we get $c(0,3) = n + 1 + \min(j, n + 1 - j)$; so we are off from the real lower bound only by lower order terms. Notice also that both heuristic algorithms considered here would choose as cutting point $\rho_1 = j$ if $j > \lceil n/2 \rceil$ and $\rho_2 = j$ if $j \leq \lceil n/2 \rceil$. Hence, they give a cost

$$\hat{c}(0,3) = \left\{\begin{matrix} n+1+n+1-j, & \text{if } j > \lceil n/2 \rceil \\ n+1+j, & \text{if } j \leq \lceil n/2 \rceil \end{matrix}\right\} = n + 1 + \min(j, n+1-j)$$

Also, by design, optimized chunksort has optimal performance when the number of items that do not have to be sorted, $\overline{m}$, is very small, since it behaves exactly as an optimal quicksort.

Very related to our work is the recent paper by Kaligosi et al. [4], where the authors consider the problem of selecting multiple ranks optimally. They propose near optimal algorithms up to $O(n)$ comparisons for the worst- and expected case of multiple selection. They make all partitions around pivots that are close to the median of the current subarray; we have shown that it is worth choosing some endpoint and then a pivot that will land very close to that endpoint. In this way, the gap between the lower and upper bound can be reduced to $n + o(n)$, and as we have seen, it might be even smaller, that is, $o(n)$.

To conclude, interval sorting provides an unified framework to study closely related problems in sorting and selection. To begin with, the results for chunksort generalize the previous results for quicksort, quickselect, partial quicksort and multiple quickselect. Considerations about optimality give rise to interesting

findings, mathematical challenges and conjectures. For instance, in the light of the iron bar cutting analogy, the optimal choice of endpoints explains the optimal strategy for selection [7]: if the rank of the sought element j is $\geq n/2$ then we should choose a pivot landing slightly to the left of j; for that, we must use a sample of $s = \Theta(\sqrt{n})$ elements and choose the element of rank $j/n \cdot s - \Delta$ from the sample, to guarantee with high probability that the pivot will land at j', very close, but to the left of j. This uses $n + o(n)$ comparisons and in the next recursive call we will use $n - j' + o(n - j') = n - j + o(n - j)$ comparisons to put a pivot slightly to the right of j, now by having picked the element of rank $\frac{j-j'}{n-j'} \cdot s + \Delta'$ from the sample. With the two pivots in place, the rest of the steps needed to select the j-th will only need $o(n)$ comparisons.

References

1. Blum, M., Floyd, R., Pratt, V., Rivest, R., Tarjan, R.: Time bounds for selection. J. Comp. Syst. Sci. 7, 448–461 (1973)
2. Hoare, C.: Find (Algorithm 65). Comm. ACM 4, 321–322 (1961)
3. Hoare, C.: Quicksort. Computer Journal 5, 10–15 (1962)
4. Kaligosi, K., Mehlhorn, K., Munro, J., Sanders, P.: Towards optimal multiple selection. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 103–114. Springer, Heidelberg (2005)
5. Kuba, M.: On quickselect, partial sorting and multiple quickselect. Inform. Process. Lett. 99(5), 181–186 (2006)
6. Martínez, C.: Partial quicksort. In: Arge, L., Italiano, G., Sedgewick, R. (eds.) Proc. of the 6th ACM-SIAM Workshop on Algorithm Engineering and Experiments (ALENEX) and the 1st ACM-SIAM Workshop on Analytic Algorithmics and Combinatorics (ANALCO), pp. 224–228 (2004)
7. Martínez, C., Panario, D., Viola, A.: Adaptive sampling for quickselect. In: Proc. of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 440–448 (2004)
8. Martínez, C., Roura, S.: Optimal sampling strategies in quicksort and quickselect. SIAM J. Comput. 31(3), 683–705 (2001)
9. Mehlhorn, K.: Data Structures and Efficient Algorithms. Sorting and Searching, vol. 1. Springer, Heidelberg (1984)
10. Prodinger, H.: Multiple Quickselect — Hoare's Find algorithm for several elements. Inform. Process. Lett. 56(3), 123–129 (1995)
11. Yao, F.: Efficient dynamic programming using quadrangle inequalities. In: Proc. of the 12th Annual ACM Symposium on the Theory of Computing (STOC), pp. 429–435 (1980)

Inapproximability of Hypergraph Vertex Cover and Applications to Scheduling Problems

Nikhil Bansal[1] and Subhash Khot[2,⋆]

[1] IBM T.J. Watson Research Center, Yorktown Heights, New York
[2] Computer Science Dept., New York University, New York

Abstract. Assuming the Unique Games Conjecture (UGC), we show optimal inapproximability results for two classic scheduling problems. We obtain a hardness of $2-\varepsilon$ for the problem of minimizing the total weighted completion time in concurrent open shops. We also obtain a hardness of $2-\varepsilon$ for minimizing the makespan in the assembly line problem.

These results follow from a new inapproximability result for the Vertex Cover problem on k-uniform hypergraphs that is stronger and simpler than previous results. We show that assuming the UGC, for every $k \geq 2$, the problem is inapproximable within $k-\varepsilon$ even when the hypergraph is *almost k-partite*.

1 Introduction

A k-uniform hypergraph $G=(V,E)$ consists of a collection of vertices $v \in V$ and a collection of edges $e \in E$ where an edge corresponds to a k element subset of V. A vertex cover of G is a subset S of vertices such that each edge contains at least one vertex from S (i.e. $e \cap S \neq \emptyset$ for each edge $e \in E$). The Ek-Vertex-Cover problem is to find the minimum vertex cover of G. This is equivalent to the Set Cover problem (by viewing edges as elements and vertices as sets) where each element lies in exactly k sets. An independent set in a k-uniform hypergraph is defined as the complement of a vertex cover, i.e. an independent set is a subset of vertices T such that no edge is completely contained in T.

The Ek-Vertex-Cover problem, and in particular the case $k=2$ corresponding to the Vertex Cover problem on graphs, is extremely well-studied. There is an almost trivial k-approximation by finding a maximal matching and the best known algorithms achieve only a slight improvement of $(1-o(1))k$ [9,13]. On the inapproximability side, for the case $k=2$, Dinur and Safra [5] obtained 1.36 NP-hardness improving on the $\frac{7}{6}-\varepsilon$ hardness by Håstad [10]. For larger k, a sequence of works obtained hardness factor $k^{1/19}$ (Trevisan [22]), $2-\varepsilon$ for some constant k (Goldreich [8]), $2-\varepsilon$ for $k=4$, $\frac{3}{2}-\varepsilon$ for $k=3$, $k^{1-o(1)}$ (Holmerin [12,11]), $k-3-\varepsilon$ (Dinur et al [3]), and finally $k-1-\varepsilon$ (Dinur et al [4]) that superseded all earlier results for $k \geq 3$.

⋆ Research supported by NSF CAREER grant CCF-0833228, NSF Expeditions grant CCF-0832795, and BSF grant 2008059.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 250–261, 2010.

Assuming the Unique Games Conjecture [14] however, optimal $k-\varepsilon$ hardness is known due to Khot and Regev [15]. Recently, in [1], the authors of this paper strengthened the Khot-Regev result for the case $k = 2$: they showed that assuming the UGC, vertex cover in graphs is $2-\varepsilon$ hard to approximate even when the graph is almost bipartite, i.e. when the graph has two disjoint independent sets $V_1, V_2 \subseteq V$ of size $(\frac{1}{2} - \varepsilon)|V|$ each. The techniques developed were also used to show a tight $2-\varepsilon$ inapproximability result (assuming a variant of the UGC) for the classic problem of minimizing the weighted completion time with precedence constraints. They designed a PCP with one free bit, near-perfect completeness, and arbitrarily low soundness, and used the equivalence between hardness of vertex cover problem and existence of PCPs with zero free bits [6,2].

In this paper, we obtain a stronger UGC-based inapproximability result for Ek-Vertex-Cover which subsumes both [15,1] and holds on hypergraphs that are almost k-partite (essential for the applications). Our result is conceptually different and much simpler than both these results (albeit using a recent invariance style theorem of Mossel [19]). Formal statement of our result is:

Theorem 1. *Assuming the Unique Games Conjecture, for any integer $k \geq 2$ and arbitrary constants $\varepsilon, \delta > 0$, given a k-uniform hypergraph $G = (V, E)$, distinguishing between the following cases is NP-hard:*

- *(YES Case): there exist disjoint subsets $V_1, \ldots, V_k \subseteq V$, satisfying $|V_i| \geq \frac{1-\varepsilon}{k}|V|$ and such that no hyperedge contains two or more vertices from some V_i.*
- *(NO Case): every vertex cover has size at least $(1-\delta)|V|$.*

Note that in the YES case, letting $V' = V \setminus (V_1 \cup \ldots \cup V_k)$, the sets $V' \cup V_i$ for $i = 1, \ldots, k$ are almost disjoint vertex covers of size at most $(\frac{1}{k} + \varepsilon)|V|$ each.

1.1 Applications

Our hypergraph vertex cover result was motivated by showing hardness results for two classic problems in scheduling. The first is the problem of minimizing weighted completion time of jobs in the so-called Concurrent Open Shop model (also referred to as the Order Scheduling model). The second is the problem of minimizing the makespan in the so-called assembly line problem.

Concurrent Open Shop: There is a collection of machines $M = \{1, \ldots, m\}$ with each machine capable of producing its own unique type of component. There is a collection of jobs $J = \{1, \ldots, n\}$ where each job requires a specific amount of components from each machine. Formally, each job $j \in J$ has weight w_j and requires a processing of p_{ij} units for each $i = 1, \ldots, m$. In the concurrent open shop model, there is no restriction on how the p_{ij} units for a job must be scheduled. The components of a job are independent of each other, and for example, all of them can be scheduled in parallel. A job is completed when all its components are completed and the goal is to minimize the total weighted completion time of the jobs.

Given its generality, the problem is widely used to model various applications in manufacturing and supply chain (see for example [16] for a survey). Several 2 approximations, using various different techniques are known for it [17,7,18]. However, until recently only NP-Hardness was known. Garg et al showed that the problem is APX-Hard [7]. More recently, Mastrolilli et al [18] showed an NP-hardness of $\frac{6}{5} - \varepsilon$, and a hardness of $\frac{4}{3} - \varepsilon$ assuming the UGC. In this work we give a tight hardness result of $2 - \varepsilon$ assuming the UGC. Our result is based on combining the construction of [18] with Theorem 1.

The Assembly Line Problem: There are $m + 1$ machines $M = \{0, \ldots, m\}$. Machine 0 is special and called the assembly machine. Each job j has an operation O_{ij} of size p_{ij} for each machine $i = 1, \ldots, m$. These operations can be done in any order. After all these operations are completed, a final operation O_{0j} of size p_{0j} is performed on machine 0 (that assembles the operations $O_{1j}, \ldots, O_{mj}$ to produce the final product j). The job j is completed when O_{0j} is completed. The goal is to minimize the makespan, or equivalently, the latest completion time of a job.

The problem is a classic one [20] and has a trivial 2-approximation: in fact any arbitrary schedule that does not waste time unnecessarily is a 2-approximation. This is because $\max_{i=0,\ldots,m} \text{Load}(i)$ is a lower bound for the problem, where $\text{Load}(i)$ is the total size of operations on machine i. On the other hand, any non-idling schedule achieves a makespan of $\text{Load}(0) + \max_{i=1,\ldots,m} \text{Load}(m)$. A somewhat better $2 - \frac{1}{m}$ approximation is also known [20]. The problem was known to be strongly NP-hard even for $m = 2$ [20]. Resolving its approximability status featured in Schuurman and Woeginger's [21] well-known list of important open problems in scheduling. It was subsequently observed by several researchers (but unpublished) that an APX-Hardness follows from known hardness results for coloring k-colorable graphs. However, these results only yield a small hardness factor. In this work we show that the trivial 2-approximation algorithm described above is essentially the best assuming the UGC.

1.2 Techniques

Our hardness reduction for the hypergraph vertex cover problem is quite simple and described in Sections 2.2, 3, and 4. Here we give a quick overview. As is standard in UGC-based reductions, the crux of our construction is a *dictatorship test*. This can be viewed as a construction of a k-uniform hypergraph with vertex set Ω^n for a finite set Ω. The hypergraph has these two properties:

- (Completeness): If the hypergraph is partitioned into $|\Omega|$ sets $\{V_\omega\}_{\omega \in \Omega}$ based on j^{th} co-ordinate for any fixed j, then each hyperedge contains at most one vertex from V_ω for any $\omega \in \Omega$.
- (Soundness): Any large (i.e. linear sized) independent set in the hypergraph must have an *influential co-ordinate*. This follows directly from a theorem of Mossel [19] that gives gaussian bounds for noise correlation of functions.

Since in the completeness case, we need the hypergraph to be (almost) k-partite, it would be natural to take $|\Omega| = k$. To satisfy the completeness property, we are

led to define $(x^1, \ldots, x^k) \in (\Omega^n)^k$ to be an hyperedge iff for every co-ordinate j, $(x^1_j, \ldots, x^k_j)$ are distinct, i.e. this tuple is a permutation of Ω. However, this does not quite work as the set of all permutations of Ω is *disconnected* as a subset of Ω^k, whereas application of Mossel's theorem has a certain connectedness requirement. We get around this problem by a simple trick that seems quite useful for future applications: we introduce a dummy element called 0 and let $\Omega = \{0, 1, \ldots, k\}$. The hyperedges are defined as before: $(x^1, \ldots, x^k) \in (\Omega^n)^k$ is an hyperedge iff for every co-ordinate j, $(x^1_j, \ldots, x^k_j)$ are distinct. Now the set of all k-tuples of distinct elements of the $k+1$ sized set Ω happens to be a connected subset of Ω^k and therefore Mossel's theorem is applicable. We let the element 0 to have negligible probability mass relative to the other elements so that the hypergraph is almost k-partite. Construction of this hypergraph (i.e. the dictatorship test) leads to a UGC-based hardness result in a straightforward manner.

2 Preliminaries

2.1 The Unique Games Conjecture

Definition 2. *An instance* $\mathcal{L}(G(V, W, E), [n], \{\sigma(v, w)\}_{(v,w)\in E})$ *of* Unique Games *consists of a regular bipartite graph* $G(V, W, E)$ *and a set* $[n]$ *of* labels. *For each edge* $(v, w) \in E$ *there is a constraint specified by a permutation* $\sigma(v, w) : [n] \mapsto [n]$. *The goal is to find a* labelling $\ell : V \cup W \to [n]$ *of the vertices such that as many edges as possible are satisfied, where an edge* $e = (v, w)$ *is said to be satisfied if* $\ell(v) = \sigma(v, w)(\ell(w))$.

Definition 3. *Given a Unique Games instance* $\mathcal{L}(G(V, W, E), [n], \{\sigma_{v,w}\}_{(v,w)\in E})$ *let* $\mathrm{OPT}(\mathcal{L})$ *denote the maximum fraction of simultaneously satisfied edges of* $\mathcal{L}$ *by any labeling, i.e.*

$$\mathrm{OPT}(\mathcal{L}) := \frac{1}{|E|} \max_{\ell : V \cup W \to [n]} |\{\, e \in E \,:\, \ell \text{ satisfies } e \,\}|.$$

A formal statement of the Unique Games Conjecture appears below; the conjecture has been widely employed to prove strong inapproximability results for several problems.

Conjecture 1 ([14]). For any constants $\zeta, \gamma > 0$, there is a sufficiently large constant $n = n(\zeta, \gamma)$ such that, for Unique Games instances $\mathcal{L}$ with label set $[n]$ it is NP-hard to distinguish between

- $\mathrm{OPT}(\mathcal{L}) \geq 1 - \zeta$,
- $\mathrm{OPT}(\mathcal{L}) \leq \gamma$.

2.2 Gaussian Bounds on Correlated Spaces

Let (Ω^k, P) be a (so-called correlated) probability space where Ω is a finite set and P is a probability distribution over Ω^k, and let the relation $R := \{x \in$

$\Omega^k \mid \mathrm{P}(x) > 0\} \subseteq \Omega^k$ be the support of P. Assume moreover that the marginal of P on any co-ordinate is the same, denote it by P as well. We say that R is a *connected relation* if for any $x, y \in R$, there exists a path $x = y(0), y(1), \ldots, y(r) = y$ in R such that $y(i)$ and $y(i+1)$ differ in one coordinate only.

Consider a k-uniform hypergraph with the vertex set Ω^n. A tuple $(x^1, \ldots, x^k) \in (\Omega^n)^k$ is defined to be an hyperedge if for every co-ordinate $1 \leq i \leq n$, we have $(x_i^1, \ldots, x_i^k) \in R$. In fact we define a probability distribution on the set of hyperedges where a random hyperedge $(x^1, \ldots, x^k)$ is selected by selecting independently for $1 \leq i \leq n$, the tuple $(x_i^1, \ldots, x_i^k)$ from space (Ω^k, P). We will use a theorem of Mossel [19] stating that every *large* independent set in this hypergraph must have an *influential co-ordinate*. Specifically, let $A \subseteq \Omega^n$ be a set and $f_A : \Omega^n \mapsto \{0,1\}$ be its indicator function. The fraction of hyperedges that lie entirely inside A is clearly,

$$\mathbb{E}_{(x^1,\ldots,x^k)} \left[\prod_{j=1}^{k} f_A(x^j) \right].$$

Mossel's theorem states that if $\mathbb{E}[f_A]$ is *large* and all influences of f_A are sufficiently small, then this expectation is lower bounded by a constant, and therefore A cannot be an independent set. In fact, as is necessary for our application, the theorem works even for (non-boolean) functions $f : \Omega^n \mapsto [0,1]$ and one only requires that *degree d influences* are sufficiently small. We state the theorem below and also define the notion of influence of variable.

Theorem 4. (Mossel [19]) *Let (Ω^k, P) be a correlated space such that the support $R \subseteq \Omega^k$ of P is connected and the minimum probability of any atom in R is at least $\alpha \leq \frac{1}{2}$. Then there exist continuous functions $\underline{\Gamma} : (0,1) \mapsto (0,1)$ and $\overline{\Gamma} : (0,1) \mapsto (0,1)$ such that the following holds: for every $\varepsilon > 0$, there exists a $\tau > 0$ and integer d such that for any function $f : \Omega^n \mapsto [0,1]$, satisfying*

$$\forall\ 1 \leq i \leq n, \quad \mathsf{Infl}_i^{\leq d}(f) \leq \tau,$$

we have:

$$\underline{\Gamma}(\mathbb{E}[f]) - \varepsilon \quad \leq \quad \mathbb{E}_{(x^1,\ldots,x^k)} \left[\prod_{j=1}^{k} f(x^j) \right] \quad \leq \quad \overline{\Gamma}(\mathbb{E}[f]) + \varepsilon.$$

Remark 1. We have stated Theorem 4 in the form that we need (we only need the lower bound). This is essentially the same statement as Theorem 6.4 in [19]. Therein, the theorem is stated for a multi-function version whereas we only need the single-function version. The bound $\rho(\Omega_i^{(1)}, \ldots, \Omega_i^{(k)}) \leq \rho < 1$ required there follows from the fact that the relation R is connected, see Lemma 2.9 therein. The author also gives quantitative bounds for $\underline{\Gamma}(), \overline{\Gamma}(), \tau$, but we don't necessarily need specific bounds.

For the sake of quick reference, we note the definition of influence and low degree influence (see [19, Section 3]). We assume that (Ω, P) is a finite probability space

and distribution on Ω^n is the product distribution P^n. Let $f = \sum_\phi \hat{f}(\phi) X_\phi$ be the multi-linear representation of f (analogue of the standard Fourier representation).

Definition 5. *For a function $f : \Omega^n \mapsto \mathbb{R}$, and a co-ordinate $1 \leq i \leq n$,*

$$\mathsf{Infl}_i(f) := \mathbb{E}[\mathrm{Var(f)}|x_1, x_{i-1}, x_{i+1}, \ldots, x_n] = \sum_{\phi:\phi_i \neq 0} \hat{f}(\phi)^2.$$

Definition 6. *For a function $f : \Omega^n \mapsto \mathbb{R}$, a co-ordinate $1 \leq i \leq n$, and integer d, the degree d influence is defined as:*

$$\mathsf{Infl}_i^{\leq d}(f) := \sum_{\phi:\phi_i \neq 0, |\phi| \leq d} \hat{f}(\phi)^2.$$

Lemma 1. *If $f : \Omega^n \mapsto [0,1]$, then the number of co-ordinates i such that $\mathsf{Infl}_i^{\leq d}(f) \geq \tau$ is at most d/τ.*

Proof. This follows from:

$$\sum_{i=1}^{n} \mathsf{Infl}_i^{\leq d}(f) = \sum_{i=1}^{n} \sum_{\phi:\phi_i \neq 0, |\phi| \leq d} \hat{f}(\phi)^2 \leq d \cdot \sum_{\phi} \hat{f}(\phi)^2 = d \cdot \|f\|_2^2 \leq d.$$

3 Dictatorship Test

In this section we define the appropriate dictatorship test that we will later use as a gadget to show the UGC based Hardness. Let Ω be a finite set consisting of $k+1$ elements, $\Omega := \{0, 1, \ldots, k\}$ (the element 0 is special as we see next). Let $\tilde{P}$ be a distribution on Ω where the elements $1, \ldots, k$ have probability mass $(1-\delta)/k$ and element 0 has mass δ. Later we will set $\delta \ll 1/k$.

Let R denote the following relation on Ω^k: A k-tuple $(a^1, \ldots, a^k) \in R$ if and only if elements $\{a^1, \ldots, a^k\}$ are all distinct. It is easy to see that one can define a probability distribution P on R such that on each co-ordinate its marginal is $\tilde{P}$ and moreover each atom in R has probability mass at least $\alpha > 0$.

Claim. R is a connected relation.

Proof. Suppose $(a^1, \ldots, a^k), (b^1, \ldots, b^k) \in R$. Let us first consider the case when $|\{a^1, \ldots, a^k, b^1, \ldots, b^k\}| = k+1$. Then there is some index i such that $b^i \notin \{a^1, \ldots, a^k\}$. We set $a^i = b^i$ and proceed inductively on the remaining $k-1$ coordinates (without ever considering coordinate i again). Now suppose that $|\{a^1, \ldots, a^k, b^1, \ldots, b^k\}| = k$. In this case we arbitrarily set some coordinate a^i to $[k+1] \setminus \{b^1, \ldots, b^k\}$, and reduce to the previous case.

Consider a hypergraph $G = (V, E)$ defined on Ω^n as follows: Each element $a \in \Omega^n$ corresponds to a vertex in G. A k-tuple $(a^1, \ldots, a^k) \in (\Omega^n)^k$ is an hyperedge if and only if the relation R is satisfied for each coordinate j, i.e. for each $1 \leq j \leq n$, $(a_j^1, a_j^2, \ldots, a_j^k) \in R$. The instance satisfies the following completeness and soundness properties:

Completeness: Fix any coordinate j, and consider the partition of the vertices based on the coordinate. That is, for $i = 0, \ldots, k$, let V_i denote the set of vertices $V_i = \{v = (a_1, \ldots, a_n) : a_j = i\}$. Clearly, $|V_0| = \delta|V|$ and $V_i = (1-\delta)|V|/k$ for $i = 1, \ldots, k$.

Claim. Any hyperedge contains at most one vertex from any set V_i.

Proof. Consider any hyperedge $e = (a^1, \ldots, a^k)$ and consider the k-tuple $(a^1_j, \ldots, a^k_j)$ of the j-th coordinates of the vertices in e. By definition, this tuple satisfies R, and hence no two of them can be i, and hence lie in V_i.

Soundness:

Claim. For every $\beta > 0$, there exists $\tau > 0$ and integer d, such that if $A \subseteq \Omega^n$ is an independent set of size $\beta|V|$, then A (or rather its indicator function) has a coordinate with degree d influence at least τ.

Proof. This follows by a direct application of Theorem 4, by choosing $\varepsilon = \underline{\Gamma}(\beta)/2$.

4 UGC Based Hardness

We assume the following variant of Unique Games Conjecture.

Hypothesis 7. *For arbitrarily small constants $\zeta, \gamma > 0$, there exists an integer $n = n(\zeta, \gamma)$ such that for a Unique Games instance $\mathcal{L}(G(V, W, E), [n], \{\sigma_{v,w}\}_{(v,w)\in E})$, it is* NP*-hard to distinguish between:*

- *(YES Case:) There is a set $W' \subseteq W$ such that $|W'| \geq (1-\zeta)|W|$ and a labeling $\ell : V \cup W \mapsto [n]$ that satisfies* every *edge (v, w) for $v \in V$ and $w \in W'$.*
- *(NO Case:) No labeling satisfies even a γ fraction of edges.*

Moreover, we can assume that the graph is both left and right regular. This variant is equivalent to the original Unique Games Conjecture as shown by Khot and Regev [15, Lemma 3.6].

Given the UGC instance $\mathcal{L}$, we construct the following hypergraph instance $H = (X, Y)$. With each vertex $v \in V$ and $w \in W$, we associate the hypercube Ω^n, where Ω is chosen as in the gadget in section 3. Let $\Omega^n(v)$ denote the hypercube for vertex v. The set of vertices X for the hypergraph is defined as $X := \cup_{w\in W} \Omega^n(w)$.

Before we describe the edges, we give some notation. Let $\sigma : [n] \to [n]$ be a permutation. For an element $x = (x_1, \ldots, x_n) \in \Omega^n$, let $x \circ \sigma$ denote the string $(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$. For each vertex $v \in V$ and every subset of its k neighbors $w^1, \ldots, w^k \in W$, we define a block $B(v, w^1, \ldots, w^k)$. There will be a collection of hyperedges for each block defined as follows: For each $x^1, x^2, \ldots, x^k \in \Omega^n(v)$ such that $(x^1, \ldots, x^k) \in R^n$, we have a hyperedge $y = (x^1 \circ \sigma(v, w^1), \ldots, x^k \circ \sigma(v, w^k))$. Here R is the relation defined in the gadget in section 3.

In other words, we identify the coordinates of $\Omega^n(w^1), \ldots, \Omega^n(w^k)$ via the permutations $\sigma(v, w^j)$, and add an hyperedge among k vertices if they satisfy the relation R on each coordinate.

Completeness: Consider the labeling ℓ in the YES case, and let W' be the set of vertices satisfying the condition in the UGC Hypothesis 7.

For $i = 0, 1, \ldots, k$, we define

$$X_i = \bigcup_{w \in W'} \{x : x \in \Omega^n(w), x_{\ell(w)} = i\}.$$

Note that for each hypercube corresponding to $w \in W'$, we are partitioning its vertices based on their $\ell(w)$ coordinate.

Claim. The sets $X_0, X_1, \ldots, X_k$ are pairwise disjoint. $|X_i| \geq (1-\delta)(1-\zeta)|X|/k$ for each $i = 1, \ldots, k$ and no hyperedge contains more than one vertex from X_i.

Proof. The disjointness of the X_i follows by construction. Since the elements $1, \ldots, k$ have probability mass $(1-\delta)/k$ in Ω, it follows that $|X_i \cap \Omega^n(w)| = (1-\delta)|\Omega^n(w)|/k$ for each $w \in W'$ and $i = 1, \ldots, k$. As $|W'| \geq (1-\zeta)|W|$ this implies the claimed bound on the size of X_i.

It remains to show that no hyperedge contains two or more vertices from X_i. Consider some hyperedge y and suppose it lies in some block $B(v, w^1, \ldots, w^k)$, then $y = (x^1 \circ \sigma(v, w^1), \ldots, x^k \circ \sigma(v, w^k))$ for some $(x^1, \ldots, x^k) \in R^n$. Let S denote the set of indices j such that $w^j \in W'$.

Since $X_i \cap \Omega^n(w) = \emptyset$ for $w \notin W'$, it follows that if $j \notin S$ then the vertex $x^j \circ \sigma(v, w^j)$ cannot lie in X_i. Thus we need to consider only the indices in S. Now for any $j \in S$, $x^j \circ \sigma(v, w^j)$ lies in X_i iff $(x^j \circ \sigma(v, w^j))_{\ell(w^j)} = i$. Since the labeling ℓ satisfies the constraints (v, w^j) for $j \in S$, we have $\sigma(v, w^j)\ell(w^j) = \ell(v)$. Thus $(x^j \circ \sigma(v, w^j))_{\ell(w^j)} = (x^j)_{\ell(v)}$ for $j \in S$. So if there exist $j \neq j' \in S$ such that both $x^j \circ \sigma(v, w^j)$ and $x^{j'} \circ \sigma(v, w^{j'})$ lie in X_i, then this implies that $(x^j)_{\ell(v)} = (x^{j'})_{\ell(v)}$ contradicting the fact that $(x^1, \ldots, x^k) \in R^n$.

Soundness: Suppose A is an independent set and that $|A| \geq \beta|X|$. Let $A_w = \Omega^n(w) \cap A$ be the vertices of A that lie in $\Omega^n(w)$ for $w \in W$. Define the boolean function $f_w : \Omega^n(w) \to \{0, 1\}$ where $f_w(x) = 1$ if $x \in A_w$ and is 0 otherwise. Let $N(v) \subseteq W$ denote the set of neighbors of $v \in V$. Let

$$f_v(x) = \mathbb{E}_{w \in N(v)}[f_w(x \circ \sigma(v, w))].$$

Since the unique games graph is regular, $\mathbb{E}_{v,x}[f_v(x)] \geq \beta$. Call a vertex $v \in V$ good if $\mathbb{E}_x[f_v(x)] \geq \beta/2$. By an averaging argument, at least $\beta/2$ fraction of vertices in V are good.

Since A is independent set, for any hyperedge $y = (x^1, \ldots, x^k)$ in block $B(v, w_1, \ldots, w_k)$, it must be that not all $f_{w^i}(x^i)$ are 1. Thus for any $v \in V$ and any k neighbors $w^1, \ldots, w^k$ of v, we have

$$\mathbb{E}_{(x^1, \ldots, x^k)} \left[\prod_{i=1}^{k} f_{w^i}(x^i \circ \sigma(v, w^i) \right] = 0. \tag{1}$$

Averaging over all k-tuples $w^1, \ldots, w^k$ of neighbors of v, we have

$$\begin{aligned}
&\mathbb{E}_{(x^1,\ldots,x^k)}\left[\prod_{i=1}^{k} f_v(x^i)\right] \\
&= \mathbb{E}_{(x^1,\ldots,x^k)}\mathbb{E}_{w^1,\ldots,w^k \in N(v)}\left[\prod_{i=1}^{k} f_{w^i}(x^i \circ \sigma(v,w^i))\right] \\
&= 0. \qquad (2)
\end{aligned}$$

Applying Theorem 4 with $\varepsilon = \underline{\Gamma}(\beta/2)/2$, it follows that f_v must have a co-ordinate with degree d influence at least τ. Let $S(v) \neq \emptyset$ be the set of all such influential co-ordinates.

For every good vertex $v \in V$ we give it an arbitrary label j from its (non-empty) set $S(v)$. For any $j \in S(v)$,

$$\begin{aligned}
\tau \leq \mathsf{Infl}_j^{\leq d}(f_v) &= \sum_{\phi_j \neq 0, |\phi| \leq d} \widehat{f_v}(\phi)^2 = \sum_{\phi_j \neq 0, |\phi| \leq d} \left(\mathbb{E}_w\left[\widehat{f_w}(\sigma(v,w)^{-1}(\phi))\right]\right)^2 \\
&\leq \sum_{\phi_j \neq 0, |\phi| \leq d} \mathbb{E}_w\left[\widehat{f_w}(\sigma(v,w)^{-1}(\phi))^2\right] = \mathbb{E}_w\left[\mathsf{Infl}_{\sigma(v,w)^{-1}(j)}^{\leq d}(f_w)\right]. \qquad (3)
\end{aligned}$$

For every $w \in W$ define the set of candidate labels for w to be

$$\mathrm{Cand}[w] = \{i \in [n] : \mathsf{Infl}_i^{\leq d} \geq \tau/2\}.$$

Since the sum of degree d influences is at most d, the number of candidates $|\mathrm{Cand}[w]|$ is at most $2d/\tau$. Since at least $\beta/2$ fraction of vertices $v \in V$ are good, and by (3) for each good vertex at least $\tau/2$ fraction of its neighbors in W have $\mathsf{Infl}_{\sigma(v,w)^{-1}(j)}^{\leq d}(f_w) \geq \tau/2$. Now if we label each vertex $w \in W$ by choosing a random element of $\mathrm{Cand}[w]$ (or any label if this set is empty), it follows that among the set of edges adjacent to good vertices v, at least a $(\tau/2)(\tau/2d)$-fraction are satisfied in expectation. Thus it follows that there is a labeling for all vertices which satisfies a $(\beta/2)(\tau/2)(\tau/2d)$ fraction of all edges. This completes the proof of soundness.

5 Applications

5.1 Total Weighted Completion Time Concurrent Open Shop

Recall the problem definition from Section 1.1. We give a reduction from Ek-Vertex-Cover. Given a k-uniform hypergraph instance $G = (V, E)$ we construct an instance I of the concurrent open shop problem as follows: A vertex $v \in V$ corresponds to a job J_v in I. Each hyperedge $e \in E$ corresponds to a machine m_e in I. Let E_v denote the set of hyperedges in G that contain the vertex v. For each edge in $e \in E_v$ we introduce an operation of unit size for job J_v that needs

to be executed on machine m_e. All other operations of J_v have size 0. Each job has weight 1.

We claim that in the YES case, the optimum average weighted completion time is at most $(1+\varepsilon)(k+1)/2$. Let $V_1, \ldots, V_k$ be the disjoint subsets of V satisfying the properties stated in theorem 1. Consider the following schedule. For each job $v \in V_i$ schedule all its unit size operations at time i (the size 0 operations can all be completed instantaneously at the beginning of the schedule, and do not matter). This gives a valid schedule for the jobs corresponding to vertices in $V_1 \cup \ldots \cup V_k$, because any hyperedge $e \in E$ contains at most one vertex from V_i, and hence no two operations are assigned to a machine at the same time. For the remaining jobs in $V \setminus (V_1 \cup \ldots \cup V_k)$, we place their operations arbitrarily in the gaps in the schedule. Since it is a k-uniform hypergraph, each machine has load at most k, and hence each job completes by time k. Thus the average completion time is at most

$$\frac{1-\varepsilon}{k}(1+2+\ldots+k)+\varepsilon k = (1-\varepsilon)(k+1)/2+\varepsilon k \leq (1+\varepsilon)(k+1)/2.$$

In the NO case, in any schedule consider the collection V' of jobs (vertices) that complete on or before time $k-1$. Now V' forms an independent set, since for any machine (hyperedge) e at most $k-1$ unit size operations could have been completed by time $k-1$, and hence at most $k-1$ vertices from e could lie in V'. Thus, if the average completion time is less than $(1-\delta)k$, this would imply that at least δ fraction of the jobs finish by time $k-1$ or sooner, which implies the existence of an independent set of size $\delta|V|$ in G.

Choosing k large enough and ε, δ arbitrarily small, by Theorem 1 it follows that the problem is hard to approximate within a factor arbitrarily close to 2 assuming the UGC.

5.2 Makespan Minimization in the Assembly Line Problem

Recall the problem definition from Section 1.1. We give a reduction from Ek-Vertex-Cover. Given a k-uniform hypergraph instance $G = (V, E)$ we construct an instance I of the assembly line problem as follows: A vertex $v \in V$ corresponds to a job J_v in I. A hyperedge $e \in E$ corresponds to a machine m_e in I. Let E_v denote the set of hyperedges in G that contain the vertex v. For each edge in $e \in E_v$ we introduce an operation of unit size for job J_v that must be processed on machine m_e. On machine 0, the job J_v has an operation of size $k/|V|$. All other operations are of size 0.

We claim that in the YES case, the optimum makespan is at most $k+1+\varepsilon k$. Let $V_1, \ldots, V_k$ be the disjoint collection of vertices satisfying the properties of Theorem 1. Consider the following schedule. For each job $v \in V_i$ schedule all its unit size operations (except those on machine 0) during the time slot $[i-1, i]$. For each of these jobs, we schedule their final operation on machine 0 arbitrarily during the interval $[i, i+1]$. This gives a valid schedule for jobs corresponding to vertices in $V_1 \cup \ldots \cup V_k$ because no hyperedge contains more than one vertex in any V_i and hence no two unit size operations are assigned to a machine

corresponding to an hyperedge at the same time. Since $V_i \leq (1-\varepsilon)|V|/k$, all the final operations of jobs in V_i can also be completed during $[i, i+1]$. For the remaining jobs in $V \setminus (V_1 \cup \ldots \cup V_k)$, we place their operations on machines $1, \ldots, m$ arbitrarily, and schedule their final operations on machine 0 arbitrarily after time $k+1$. Thus the makespan is at most

$$k + 1 + \varepsilon|V| \cdot k/|V| = k + 1 + \varepsilon k.$$

In the NO case, in any schedule let V' denote the subset of vertices corresponding to the jobs for which the final operation is scheduled before time $k-1$. We claim that V' corresponds to an independent set, because for any machine (hyperedge) at most $k-1$ operations of unit size could have been completed by time $k-1$, and hence at most $k-1$ vertices from the hyperedge could lie in V'. Thus, if the makespan is less than $2k - 1 - \delta k = k - 1 + (1-\delta)|V| \cdot k/|V|$, this implies that $|V'| \geq \delta|V|$ and hence there is an independent set of size $\delta|V|$.

Choosing k large enough and ε, δ arbitrarily small, by Theorem 1 it follows that the problem is hard to approximate within a factor arbitrarily close to 2 assuming the UGC.

References

1. Bansal, N., Khot, S.: An optimal long code test with one free bit. In: FOCS (2009)
2. Bellare, M., Goldreich, O., Sudan, M.: Free bits, PCPs, and nonapproximability—towards tight results. SIAM Journal on Computing 27(3), 804–915 (1998)
3. Dinur, I., Guruswami, V., Khot, S.: Vertex cover on k-uniform hypergraphs is hard to approximate within factor $(k-3-\varepsilon)$. In: Electronic Colloquium on Computational Complexity, Technical Report TR02-027 (2002)
4. Dinur, I., Guruswami, V., Khot, S., Regev, O.: A new multilayered PCP and the hardness of hypergraph vertex cover. SIAM Journal on Computing 34(5), 1129–1146 (2005)
5. Dinur, I., Safra, S.: On the hardness of approximating minimum vertex cover. Annals of Mathematics 162(1) (2005) (Preliminary version in STOC 2002)
6. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. Journal of the ACM 43(2), 268–292 (1996)
7. Garg, N., Kumar, A., Pandit, V.: Order scheduling models: Hardness and algorithms. In: FSTTCS, pp. 96–107 (2007)
8. Goldreich, O.: Using the FGLSS-reduction to prove inapproximability results for minimum vertex cover in hypergraphs. ECCC Technical Report TR01-102 (2001)
9. Halperin, E.: Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. SIAM Journal on Computing 31(5), 1608–1623 (2002)
10. Håstad, J.: Some optimal inapproximability results. Journal of ACM 48(4), 798–859 (2001)
11. Holmerin, J.: Improved inapproximability results for vertex cover on k-regular hyper-graphs. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 1005–1016. Springer, Heidelberg (2002)
12. Holmerin, J.: Vertex cover on 4-regular hyper-graphs is hard to approximate within $2-\varepsilon$. In: Proc. 34th ACM Symp. on Theory of Computing (STOC), pp. 544–552 (2002)

13. Karakostas, G.: A better approximation ratio for the vertex cover problem. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1043–1050. Springer, Heidelberg (2005)
14. Khot, S.: On the power of unique 2-prover 1-round games. In: Proc. 34th ACM Symposium on Theory of Computing (2002)
15. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within 2-epsilon. J. Comput. Syst. Sci. 74(3), 335–349 (2008)
16. Leung, J., Li, H., Pinedo, M.: Order scheduling models: an overview. In: Multidisciplinary Scheduling: Theory and Applications, pp. 37–56 (2003)
17. Leung, J.Y.T., Li, H., Pinedo, M.: Scheduling orders for multiple product types to minimize total weighted completion time. Discrete Appl. Math. 155, 945–970 (2007)
18. Mastrolilli, M., Queyranne, M., Schulz, A.S., Svensson, O., Uhan, N.A.: Minimizing the sum of weighted completion times in a concurrent open shop. Preprint (2009)
19. Mossel, E.: Gaussian bounds for noise correlation of functions. To Appear in GAFA. Current version on arxiv/math/0703683 (2009)
20. Potts, C.N., Sevastianov, S.V., Strusevish, V.A., Van Wassenhove, L.N., Zwaneveld, C.M.: The two-stage assembly scheduling problem: Complexity and approximation. Operations Research 43, 346–355 (1995)
21. Schuurman, P., Woeginger, G.J.: Polynomial time approximation algorithms for machine scheduling: ten open problems. Journal of Scheduling 2, 203–213 (1999)
22. Trevisan, L.: Non-approximability results for optimization problems on bounded degree instances. In: Proc. 33rd ACM Symp. on Theory of Computing (STOC), pp. 453–461 (2001)

Thresholded Covering Algorithms for Robust and Max-min Optimization

Anupam Gupta[1,⋆], Viswanath Nagarajan[2], and R. Ravi[3,⋆⋆]

[1] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
[3] Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract. The general problem of robust optimization is this: one of several possible scenarios will appear tomorrow and require coverage, but things are more expensive tomorrow than they are today. What should you anticipatorily buy today, so that the worst-case covering cost (summed over both days) is minimized? We consider the *k-robust model* [6,15] where the possible scenarios tomorrow are given by all demand-subsets of size k.

We present a simple and intuitive template for k-robust problems. This gives improved approximation algorithms for the k-robust Steiner tree and set cover problems, and the first approximation algorithms for k-robust Steiner forest, minimum-cut and multicut. As a by-product of our techniques, we also get approximation algorithms for *k-max-min* problems of the form: "*given a covering problem instance, which k of the elements are costliest to cover?*"

1 Introduction

Consider the following *k-robust* set cover problem: we are given a set system $(U, \mathcal{F} \subseteq 2^U)$. Tomorrow some set of k elements $S \subseteq U$ will want to be covered; however, today we don't know what this set will be. One strategy is to wait until tomorrow and buy an $O(\log n)$-approximate set cover for this set. However, sets are cheaper today: they will cost λ times as much tomorrow as they cost today. Hence, it may make sense to buy some anticipatory partial solution today (i.e. in the first-stage), and then complete it tomorrow (i.e. second-stage) once we know the actual members of the set S. Since we do not know anything about the set S (or maybe we are risk-averse), we want to plan for the worst-case, and minimize

$$(\text{cost of anticipatory solution}) + \lambda \cdot \max_{S:|S|\leq k} (\text{additional cost to cover } S).$$

Early approximation results for robust problems [4,10] had assumed that the collection of possible sets S was explicitly given (and the performance guarantee

⋆ Supported in part by NSF awards CCF-0448095 and CCF-0729022, and an Alfred P. Sloan Fellowship.
⋆⋆ Supported in part by NSF grant CCF-0728841.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 262–274, 2010.

depended logarithmically on the size of this collection). Since this seemed quite restrictive, Feige et al. [6] proposed this k-robust model where any of the $\binom{n}{k}$ subsets S of size k could arrive. Though this collection of possible sets was potentially exponential sized (for large values of k), the hope was to get results that did not depend polynomially on k. To get such an algorithm, Feige et al. considered the k-max-min set-cover problem ("which subset $S \subseteq U$ of size k requires the largest set cover value?"), and used the online algorithm for set cover to get a greedy-style $O(\log m \log n)$ approximation for this problem; here m and n are the number of sets and elements in the set system. They then used this max-min problem as a separation oracle in an LP-rounding-based algorithm (à la [20]) to get the same approximation guarantee for the k-robust problem. They also showed the max-min and k-robust set cover problems to be $\Omega(\frac{\log m}{\log \log m} + \log n)$ hard—which left a logarithmic gap between the upper and lower bounds. However, an online algorithm based approach is unlikely to close this gap, since the online algorithm for set cover is necessarily a log-factor worse that its offline counterparts [2].

Apart from the obvious goal of improving the result for this particular problem, one may want to develop algorithms for other k-robust problems. E.g., for the *k-robust min-cut* problem, some set S of k sources will want to be separated from the sink vertex tomorrow, and we want to find the best way to cut edges to minimize the total cost incurred (over the two days) for the worst-case k-set S. Similarly, in the *k-robust Steiner forest*, we are given a metric space and a collection of source-sink pairs, and any set S of k source-sink pairs may desire pairwise connection tomorrow; what should we do to minimize the sum of the costs incurred over the two days? One can extend the Feige et al. framework to first solve max-min problems using online algorithms, but in all cases we seem to lose extra logarithmic factors. Moreover, for the above two problems (and others), the LP-rounding-based framework does not seem to extend directly. The latter obstacle was also observed by Khandekar et al. [15], who gave constant-factor algorithms for k-robust versions of Steiner tree and facility location.

1.1 Main Results

In this paper, we present a general template to design algorithms for k-robust problems. We improve on previous results, by obtaining an $O(\log m + \log n)$ factor for k-robust set cover, and improving the constant in the approximation factor for Steiner tree. We also give the first algorithms for some other standard covering problems, getting constant-factor approximations for both k-robust Steiner forest—which was left open by Khandekar et al.—and for k-robust min-cut, and an $O(\frac{\log^2 n}{\log \log n})$ approximation for k-robust multicut. Our algorithms do not use a max-min subroutine directly: however, our approach ends up giving us approximation algorithms for k-max-min versions of set cover, Steiner forest, min-cut and multicut; all but the one for multicut are best possible under standard assumptions; the ones for Steiner forest and multicut are the first known algorithms for their k-max-min versions.

An important contribution of our work (even more than the new/improved approximation guarantees) is the simplicity of the algorithms, and the ideas in their analysis. The following is our actual algorithm for k-robust set cover.

> Suppose we "guess" that the maximum second-stage cost in the optimal solution is T. Let $A \subseteq U$ be all elements for which the cheapest set covering them costs more than $\beta \cdot T/k$, where $\beta = O(\log m + \log n)$. We build a set cover on A as our first stage. (Say this cover costs C_T.)

To remove the guessing, try all values of T and choose the solution that incurs the least total cost $C_T + \lambda\beta T$. Clearly, by design, no matter which k elements arrive tomorrow, it will not cost us more than $\lambda \cdot k \cdot \beta T/k = \lambda\beta T$ to cover them, which is within β of what the optimal solution pays.

The key step of our analysis is to argue why C_T is close to optimum. We briefly describe the intuition; details appear in Section 3. Suppose $C_T \gg \beta\mathsf{Opt}$: then the fractional solution to the LP for set cover for A would cost $\gg \frac{\beta}{\ln n}\mathsf{Opt} \gg \mathsf{Opt}$, and so would its dual. Our key technical contribution is to show how to *"round" this dual LP* to find a "witness" $A' \subseteq A$ with only k elements, and also a corresponding feasible dual of value $\gg \mathsf{Opt}$—i.e., the dual value does not decrease much in the rounding (This rounding uses the fact that the algorithm only put those elements in A that were expensive to cover). Using duality again, this proves that the optimal LP value, and hence the optimal set cover for these k elements A', would cost much more than Opt—a contradiction!

In fact, our algorithms for the other k-robust problems are almost identical to this one; indeed, the only slightly involved algorithm is that for k-robust Steiner forest. Of course, the proofs to bound the cost C_T need different ideas in each case. For example, directly rounding the dual for Steiner forest is difficult, so we give a primal-dual argument to show the existence of such a witness $A' \subseteq A$ of size at most k. For the cut-problems, one has to deal with additional issues because Opt consists of two stages that have to be charged to separately, and this requires a Gomory-Hu-tree-based charging. Even after this, we still have to show that if the cut for a set of sources A is large then there is a witness $A' \subseteq A$ of at most k sources for which the cut is also large—i.e., we have to somehow aggregate the flows (i.e. dual-rounding for cut problems). We prove new flow-aggregation lemmas for single-sink flows using Steiner-tree-packing results, and for multiflows using oblivious routing [18]; both proofs are possibly of independent interest.

Paper Outline. In Sections 2 and 2.1 we present the formal framework for k-robust problems, and abstract out the properties that we'd like from our algorithms. Section 3 contains such an algorithm for k-robust set cover, and k-robust min-cut appears in Section 4. We refer the reader to the full version of the paper [13] for missing proofs and results on k-robust Steiner forest and multicut. There we also present a general reduction from robust problems to the corresponding max-min problems, which has implications for uncertainty sets specified by matroid and knapsack type constraints.

1.2 Related Work

Approximation algorithms for robust optimization was initiated by Dhamdhere et al. [4]: they study the case when the scenarios were explicitly listed, and gave constant-factor approximations for Steiner tree and facility location, and logarithmic approximations to mincut/multicut problems. Golovin et al. [10] improved the mincut result to a constant factor approximation, and also gave an $O(1)$-approximation for robust shortest-paths. The k-robust model was introduced in Feige et al. [6], where they gave an $O(\log m \log n)$-approximation for set cover. Khandekar et al. [15] noted that the techniques of [6] did not give good results for Steiner tree, and developed new constant-factor approximations for k-robust versions of Steiner tree, Steiner forest on trees and facility location. Using our framework, the algorithm we get for Steiner tree can be viewed as a rephrasing of their algorithm—our proof is arguably more transparent and results in a better bound. Our approach can also be used to get a slightly better ratio than [15] for the Steiner forest problem on trees.

Constrained submodular maximization problems [17,7,23,3,25] appear very relevant at first sight: e.g., the k-max-min version of min-cut ("find the k sources whose separation from the sink costs the most") is precisely submodular maximization under a cardinality constraint, and hence is approximable to within $(1-1/e)$. But apart from min-cut, the other problems do not give us submodular functions to maximize, and massaging the functions to make them submodular seems to lose logarithmic factors. E.g., one can use tree embeddings [5] to reduce Steiner tree to a problem on trees and make it submodular; in other cases, one can use online algorithms to get submodular-like properties (we give a general reduction for covering problems that admit good offline and online algorithms in [13]). Eventually, it is unclear how to use existing results on submodular maximization in any general way.

Considering the *average* instead of the worst-case performance gives rise to the well-studied model of stochastic optimization [19,14]. Some common generalizations of the robust and stochastic models have been considered (see, e.g., Swamy [24] and Agrawal et al. [1]).

Feige et al. [6] also considered the k-max-min set cover—they gave an $O(\log m \log n)$-approximation algorithm for this, and used it in the algorithm for k-robust set cover. They also showed an $\Omega(\frac{\log m}{\log \log m})$ hardness of approximation for k-max-min (and k-robust) set cover. To the best of our knowledge, none of the k-max-min problems other than min-cut have been studied earlier.

The k-*min*-*min* versions of covering problems (i.e. "which k demands are the *cheapest* to cover?") have been extensively studied for set cover [21,8], Steiner tree [9], Steiner forest [12], min-cut and multicut [11,18]. However these problems seem to be related to the k-max-min versions only in spirit.

2 Notation and Definitions

Deterministic covering problems. A covering problem Π has a ground-set E of elements with costs $c : E \to \mathbb{R}_+$, and n covering requirements (often called

demands or clients), where the solutions to the i-th requirement is specified—possibly implicitly—by a family $\mathcal{R}_i \subseteq 2^E$ which is upwards closed (since this is a covering problem). Requirement i is *satisfied* by solution $S \subseteq E$ iff $S \in \mathcal{R}_i$. The covering problem $\Pi = \langle E, c, \{\mathcal{R}_i\}_{i=1}^n \rangle$ involves computing a solution $S \subseteq E$ satisfying all n requirements and having minimum cost $\sum_{e \in S} c_e$. E.g., in set cover, "requirements" are items to be covered, and "elements" are sets to cover them with. In Steiner tree, requirements are terminals to connect to the root and elements are the edges; in multicut, requirements are terminal pairs to be separated, and elements are edges to be cut.

Robust covering problems. This problem, denoted $\mathsf{Robust}(\Pi)$, is a *two-stage optimization* problem, where elements are possibly bought in the first stage (at the given cost) or the second stage (at cost λ times higher). In the second stage, some subset $\omega \subseteq [n]$ of requirements (also called a *scenario*) materializes, and the elements bought in both stages must satisfy each requirement in ω. Formally, the input to problem $\mathsf{Robust}(\Pi)$ consists of (a) the covering problem $\Pi = \langle E, c, \{\mathcal{R}_i\}_{i=1}^n \rangle$ as above, (b) a set $\Omega \subseteq 2^{[n]}$ of scenarios (possibly implicitly given), and (c) an inflation parameter $\lambda \geq 1$. Since we deal with covering problems, it can be assumed WLOG that the uncertainty set Ω is downwards closed. A feasible solution to $\mathsf{Robust}(\Pi)$ is a set of *first stage elements* $E_0 \subseteq E$ (bought without knowledge of the scenario), along with an *augmentation algorithm* that given any $\omega \in \Omega$ outputs $E_\omega \subseteq E$ such that $E_0 \cup E_\omega$ satisfies all requirements in ω. The objective function is to minimize: $c(E_0) + \lambda \cdot \max_{\omega \in \Omega} c(E_\omega)$. Given such a solution, $c(E_0)$ is called the first-stage cost and $\max_{\omega \in \Omega} c(E_\omega)$ is the second-stage cost.

k-robust problems. In this paper, we deal with robust covering problems under *cardinality* uncertainty sets: i.e., $\Omega := \binom{[n]}{k} = \{S \subseteq [n] \mid |S| = k\}$. We denote this problem by $\mathsf{Robust}_k(\Pi)$.

Max-min problems. Given a covering problem Π and a set Ω of scenarios, the *max-min* problem involves finding a scenario $\omega \in \Omega$ for which the cost of the min-cost solution to ω is maximized. Note that by setting $\lambda = 1$ in any robust covering problem, *the optimal value of the robust problem equals that of its corresponding max-min problem.* In a **k-max-min problem** we have $\Omega = \binom{[n]}{k}$.

2.1 The Abstract Properties We Want from Our Algorithms

Our algorithms for robust and max-min versions of covering problems are based on the following guarantee.

Definition 1. *An algorithm is* $(\alpha_1, \alpha_2, \beta)$-discriminating *iff given as input any instance of* $\mathsf{Robust}_k(\Pi)$ *and a threshold* T*, the algorithm outputs (i) a set* $\Phi_T \subseteq E$*, and (ii) an algorithm* $\mathsf{Augment}_T : \binom{[n]}{k} \to 2^E$*, such that:*

A. For every scenario $D \in \binom{[n]}{k}$*,*

(i) the elements in $\Phi_T \cup \mathsf{Augment}_T(D)$ *satisfy all requirements in* D*, and*

(ii) the resulting augmentation cost $c(\mathsf{Augment}_T(D)) \leq \beta \cdot T$*.*

B. Let Φ^* and T^* (respectively) denote the first-stage and second-stage cost of an optimal solution to the $\mathsf{Robust}_k(\Pi)$ *instance. If the threshold $T \geq T^*$ then the first stage cost $c(\Phi_T) \leq \alpha_1 \cdot \Phi^* + \alpha_2 \cdot T^*$.*

The next lemma shows why having a discriminating algorithm is sufficient to solve the robust problem. The issue to address is that having guessed T for the optimal second stage cost, we have no direct way of verifying the correctness of that guess—hence we choose the best among all possible values of T. For $T \approx T^*$ the guarantees in Definition 1 ensure that we pay $\approx \Phi^* + T^*$ in the first stage, and $\approx \lambda T^*$ in the second stage; for guesses $T \ll T^*$, the first-stage cost in guarantee (B) is likely to be large compared to Opt.

Lemma 1. *If there is an $(\alpha_1, \alpha_2, \beta)$-discriminating algorithm for a robust covering problem* $\mathsf{Robust}_k(\Pi)$*, then for every $\epsilon > 0$ there is a $\left((1+\epsilon) \cdot \max\left\{\alpha_1, \beta + \frac{\alpha_2}{\lambda}\right\}\right)$-approximation algorithm for* $\mathsf{Robust}_k(\Pi)$*.*

In the rest of the paper, we focus on providing discriminating algorithms for suitable values of $\alpha_1, \alpha_2, \beta$.

2.2 Additional Property Needed for k-Max-min Approximations

As we noted above, a k-max-min problem is a robust problem where the inflation $\lambda = 1$ (which implies that in an optimal solution $\Phi^* = 0$, and T^* is the k-max-min value). Hence a discriminating algorithm immediately gives an approximation to the *value*: for any $D \in \binom{[n]}{k}$, $\Phi_T \cup \mathsf{Augment}_T(D)$ satisfies all demands in D, and for the right guess of $T \approx T^*$, the cost is at most $(\alpha_2 + \beta)T^*$. It remains to output a bad k-set as well, and hence the following definition is useful.

Definition 2. *An algorithm for a robust problem is* strongly discriminating *if it satisfies the properties in Definition 1, and when the inflation parameter is $\lambda = 1$ (and hence $\Phi^* = 0$), the algorithm also outputs a set $Q_T \in \binom{[n]}{k}$ such that if $c(\Phi_T) \geq \alpha_2 T$, the cost of optimally covering the set Q_T is $\geq T$.*

Recall that for a covering problem Π, the cost of optimally covering the set of requirements $Q \in \binom{[n]}{k}$ is $\min\{c(E_Q) \mid E_Q \subseteq E \text{ and } E_Q \in \mathcal{R}_i \ \forall i \in Q\}$.

Lemma 2. *If there is an $(\alpha_1, \alpha_2, \beta)$-*strongly*-discriminating algorithm for a robust covering problem* $\mathsf{Robust}_k(\Pi)$*, then for every $\epsilon > 0$ there is an algorithm for k-max-min(Π) that outputs a set Q such that for some T, the optimal cost of covering this set Q is at least T, but every k-set can be covered with cost at most $(1+\epsilon) \cdot (\alpha_2 + \beta)\, T$.*

3 Set Cover

Consider the k-robust set cover problem where there is a set system $(U, \mathcal{F})$ with a universe of $|U| = n$ elements, and m sets in $\mathcal{F}$ with each set $R \in \mathcal{F}$ costing c_R, an inflation parameter λ, and an integer k such that each of the sets $\binom{U}{k}$ is a possible scenario for the second-stage. Given Lemma 1, it suffices to

Algorithm 1. Algorithm for k-Robust Set Cover

1: **input:** k-robust set-cover instance and threshold T.
2: **let** $\beta \leftarrow 36 \ln m$, and $S \leftarrow \{v \in U \mid$ min cost set covering v has cost at least $\beta \cdot \frac{T}{k}\}$.
3: **output** first stage solution Φ_T as the Greedy-Set-Cover(S).
4: **define** $\mathsf{Augment}_T(\{i\})$ as the min-cost set covering i, for $i \in U \setminus S$; and $\mathsf{Augment}_T(\{i\}) = \emptyset$ for $i \in S$.
5: **output** second stage solution $\mathsf{Augment}_T$, where $\mathsf{Augment}_T(D) := \bigcup_{i \in D} \mathsf{Augment}_T(\{i\})$ for all $D \subseteq U$.

show a discriminating algorithm as defined in Definition 1 for this problem. The algorithm given below is easy: pick all elements which can only be covered by expensive sets, and cover them in the first stage.

Claim 1 (Property A for Set Cover). *For all $T \geq 0$ and scenario $D \in \binom{U}{k}$, the sets $\Phi_T \bigcup \mathsf{Augment}_T(D)$ cover elements in D, and $c(\mathsf{Augment}_T(D)) \leq \beta T$.*

Proof: The elements in $D \cap S$ are covered by Φ_T; and by definition of $\mathsf{Augment}_T$, each element $i \in D \setminus S$ is covered by set $\mathsf{Augment}_T(\{i\})$. Thus we have the first part of the claim. For the second part, note that by definition of S, the cost of $\mathsf{Augment}_T(\{i\})$ is at most $\beta T/k$ for all $i \in U$. ■

Theorem 1 (Property B for Set Cover). *Let Φ^* denote the optimal first stage solution (and its cost), and T^* the optimal second stage cost. Let $\beta = 36 \ln m$. If $T \geq T^*$ then $c(\Phi_T) \leq H_n \cdot (\Phi^* + 12 \cdot T^*)$.*

Proof: We claim that there is a *fractional* solution $\bar{x}$ for the set covering instance S with small cost $O(\Phi^* + T^*)$, whence rounding this to an integer solution implies the theorem, since the greedy algorithm has performance ratio H_n. For a contradiction, assume not: let every fractional set cover be expensive, and hence there must be a dual solution of large value. We then *round this dual solution* to get a dual solution to a sub-instance with only k elements that costs $> \Phi^* + T^*$, which is impossible (since using the optimal solution we can solve every instance on k elements with that cost).

To this end, let $S' \subseteq S$ denote the elements that are *not* covered by the optimal first stage Φ^*, and let $\mathcal{F}' \subseteq \mathcal{F}$ denote the sets that contain at least one element from S'. By the choice of S, all sets in $\mathcal{F}'$ cost at least $\beta \cdot \frac{T}{k} \geq \beta \cdot \frac{T^*}{k}$. Define the "coarse" cost for a set $R \in \mathcal{F}'$ to be $\widehat{c}_R = \lceil \frac{c_R}{6T^*/k} \rceil$. For each set $R \in \mathcal{F}'$, since $c_R \geq \frac{\beta T^*}{k} \geq \frac{6T^*}{k}$, it follows that $\widehat{c}_R \cdot \frac{6T^*}{k} \in [c_R, 2 \cdot c_R)$, and also that $\widehat{c}_R \geq \beta/6$.

Now consider the following primal-dual pair of LPs for the set cover instance with elements S' and sets $\mathcal{F}'$ having the coarse costs $\widehat{c}$.

$$\begin{array}{ll} \min \sum_{R \in \mathcal{F}'} \widehat{c}_R \cdot x_R & \\ \sum_{R \ni e} x_R \geq 1, & \forall e \in S', \\ x_R \geq 0, & \forall R \in \mathcal{F}'. \end{array} \qquad \begin{array}{ll} \max \sum_{e \in S'} y_e & \\ \sum_{e \in R} y_e \leq \widehat{c}_R, & \forall R \in \mathcal{F}', \\ y_e \geq 0, & \forall e \in S'. \end{array}$$

Let $\{x_R\}_{R \in \mathcal{F}'}$ be an optimal primal and $\{y_e\}_{e \in S'}$ an optimal dual solution. The following claim bounds the (coarse) cost of these fractional solutions.

Claim 2. *If $\beta = 36 \ln m$, then the LP cost is $\sum_{R \in \mathcal{F}'} \widehat{c}_R \cdot x_R = \sum_{e \in S'} y_e \leq 2 \cdot k$.*

Before we prove Claim 2, let us assume it and complete the proof of Theorem 1. Given the primal LP solution $\{x_R\}_{R \in \mathcal{F}'}$ to cover elements in S', define an LP solution to cover elements in S as follows: define $z_R = 1$ if $R \in \Phi^*$, $z_R = x_R$ if $R \in \mathcal{F}' \setminus \Phi^*$; and $z_R = 0$ otherwise. Since the solution $\bar{z}$ contains Φ^* integrally, it covers elements $S \setminus S'$ (i.e. the portion of S covered by Φ^*); since $z_R \geq x_R$, $\bar{z}$ fractionally covers S'. Finally, the cost of this solution is $\sum_R c_R z_R \leq \Phi^* + \sum_R c_R x_R \leq \Phi^* + \frac{6T^*}{k} \cdot \sum_R \widehat{c}_R x_R$. But Claim 2 bounds this by $\Phi^* + 12 \cdot T^*$. Since we have a LP solution of value $\Phi^* + 12T^*$, and the greedy algorithm is an H_n-approximation relative to the LP value for set cover, this completes the proof. ■

Claim 1 and Theorem 1 show our algorithm for set cover to be an $(H_n, 12H_n, 36 \ln m)$-discriminating algorithm. Applying Lemma 1 converts this discriminating algorithm to an algorithm for k-robust set cover, and gives the following improvement to the result of [6].

Theorem 2. *There is an $O(\log m + \log n)$-approximation for k-robust set cover.*

It remains to give the proof for Claim 2 above; indeed, that is where the technical heart of the result lies.

Proof of Claim 2: Recall that we want to bound the optimal fractional set cover cost for the instance $(S', \mathcal{F}')$ with the coarse (integer) costs; x_R and y_e are the optimal primal and dual solutions. For a contradiction, assume that the LP cost $\sum_{R \in \mathcal{F}'} \widehat{c}_R x_R = \sum_{e \in S'} y_e$ lies in the unit interval $((\gamma - 1)k, \gamma k]$ for some integer $\gamma \geq 3$.

Define integer-valued random variables $\{Y_e\}_{e \in S'}$ by setting, for each $e \in S'$ independently, $Y_e = \lfloor y_e \rfloor + I_e$, where I_e is a Bernoulli$(y_e - \lfloor y_e \rfloor)$ random variable. We claim that whp the random variables $Y_e/3$ form a feasible dual— i.e., they satisfy all the constraints $\{\sum_{e \in R}(Y_e/3) \leq \widehat{c}_R\}_{R \in \mathcal{F}'}$ with high probability. Indeed, consider a dual constraint corresponding to $R \in \mathcal{F}'$: since we have $\sum_{e \in R} \lfloor y_e \rfloor \leq \widehat{c}_R$, we get that $Pr[\sum_{e \in R} Y_e > 3 \cdot \widehat{c}_R] \leq Pr[\sum_{e \in R} I_e > 2 \cdot \widehat{c}_R]$. But now we use a Chernoff bound [16] to bound the probability that the sum of independent 0-1 r.v.s, $\sum_{e \in R} I_e$, exceeds twice its mean (here $\sum_{e \in R} E[I_e] \leq \sum_{e \in R} y_e \leq \widehat{c}_R$) by $\varepsilon^{-\widehat{c}_R/3} \leq e^{-\beta/18} \leq m^{-2}$, since each $\widehat{c}_R \geq \beta/6$ and $\beta = 36 \cdot \ln m$. Finally, a trivial union bound implies that $Y_e/3$ satisfies all the m contraints with probability at least $1 - 1/m$. Moreover, the expected dual objective is $\sum_{e \in S'} y_e \geq (\gamma - 1)k \geq 1$ (since $\gamma \geq 3$ and $k \geq 1$), and by another Chernoff Bound, $\Pr[\sum_{e \in S'} Y_e > \frac{\gamma - 1}{2} \cdot k] \geq a$ (where $a > 0$ is some constant). Putting it all together, with probability at least $a - \frac{1}{m}$, we have a *feasible* dual solution $Y'_e := Y_e/3$ with objective value at least $\frac{\gamma - 1}{6} \cdot k$.

Why is this dual Y'_e any better than the original dual y_e? It is "near-integral"— specifically, each Y'_e is either zero or at least $\frac{1}{3}$. So order the elements of S' in decreasing order of their Y'-value, and let Q be the set of the *first k elements* in this order. The total dual value of elements in Q is at least $\min\{\frac{\gamma - 1}{6}k, \frac{k}{3}\} \geq \frac{k}{3}$, since $\gamma \geq 3$, and each non-zero Y' value is $\geq 1/3$. This valid dual for elements

in Q shows a lower bound of $\frac{k}{3}$ on minimum (fractional) $\widehat{c}$-cost to cover the k elements in Q. Using $c_R > \frac{3T^*}{k} \cdot \widehat{c}_R$ for each $R \in \mathcal{F}'$, the minimum c-cost to fractionally cover Q is $> \frac{3T^*}{k} \cdot \frac{k}{3} = T^*$. Hence, if Q is the realized scenario, the optimal second stage cost will be $> T^*$ (as no element in Q is covered by Φ^*)—this contradicts the fact that OPT can cover $Q \in \binom{U}{k}$ with cost at most T^*. Thus we must have $\gamma \leq 2$, which completes the proof of Claim 2. ■

The k-Max-Min Set Cover Problem. The proof of Claim 2 suggests how to get a $(H_n, 12H_n, 36 \ln m)$ strongly discriminating algorithm. When $\lambda = 1$ (and so $\Phi^* = 0$), the proof shows that if $c(\Phi_T) > 12H_n \cdot T$, there is a randomized algorithm that outputs k-set Q with optimal covering cost $> T$ (witnessed by the dual solution having cost $> T$). Now using Lemma 2, we get the claimed $O(\log m + \log n)$ algorithm for the k-max-min set cover problem. This nearly matches the hardness of $\Omega(\frac{\log m}{\log \log m} + \log n)$ given by [6].

Remarks: We note that our k-robust algorithm also extends to the more general setting of (uncapacitated) *Covering Integer Programs*; a CIP (see eg. [22]) is given by $A \in [0,1]^{n \times m}$, $b \in [1, \infty)^n$ and $c \in \mathbb{R}^m_+$, and the goal is to minimize $\{c^T \cdot x \mid Ax \geq b, x \in \mathbb{Z}^m_+\}$. The results above (as well as the [6] result) also hold in the presence of set-dependent inflation factors—details will appear in the full version. Results for the other covering problems do not extend to the case of non-uniform inflation: this is usually inherent, and not just a flaw in our analysis. Eg., [15] give an $\Omega(\log^{1/2-\epsilon} n)$ hardness for k-robust Steiner forest under just two distinct inflation-factors, whereas we give an $O(1)$-approximation under uniform inflations.

4 Minimum Cut

We now consider the k-robust minimum cut problem, where we are given an undirected graph $G = (V, E)$ with edge capacities $c : E \to \mathbb{R}_+$, a root $r \in V$, terminals $U \subseteq V$, inflation factor λ. Again, any subset in $\binom{U}{k}$ is a possible second-stage scenario, and again we seek to give a discriminating algorithm. This algorithm, like for set cover, is non-adaptive: we just pick all the "expensive" terminals and cut them in the first stage.

Algorithm 2. Algorithm for k-Robust Min-Cut

1: **input:** k-robust minimum-cut instance and threshold T.
2: **let** $\beta \leftarrow \Theta(1)$, and
$S \leftarrow \{v \in U \mid$ min cut separating v from root r costs at least $\beta \cdot \frac{T}{k}\}$.
3: **output** first stage solution Φ_T as the minimum cut separating S from r.
4: **define** $\mathsf{Augment}_T(\{i\})$ as the min-r-i cut in $G \setminus \Phi_T$, for $i \in U \setminus S$; and $\mathsf{Augment}_T(\{i\}) = \emptyset$ for $i \in S$.
5: **output** second stage solution $\mathsf{Augment}_T$, where
$\mathsf{Augment}_T(D) := \bigcup_{i \in D} \mathsf{Augment}_T(\{i\})$ for all $D \subseteq U$.

Claim 3 (Property A for Min-Cut). *For all $T \geq 0$ and $D \in \binom{U}{k}$, the edges $\Phi_T \bigcup \mathsf{Augment}_T(D)$ separate terminals D from r; and $c(\mathsf{Augment}_T(D)) \leq \beta T$.*

Theorem 3 (Property B for Min-Cut). *Let Φ^* denote the optimal first stage solution (and its cost), and T^* the optimal second stage cost. If $\beta \geq \frac{10e}{e-1}$ and $T \geq T^*$ then $c(\Phi_T) \leq 3 \cdot \Phi^* + \frac{\beta}{2} \cdot T^*$.*

Here's the intuition for this theorem: As in the set cover proof, we claim that if the optimal cost of separating S from the root r is high, then there must be a dual solution (which prescribes flows from vertices in S to r) of large value. We again "round" this dual solution by aggregating these flows to get a set of k terminals that have a large combined flow (of value $> \Phi^* + T^*$) to the root—but this is impossible, since the optimal solution promises us a cut of at most $\Phi^* + T^*$ for any set of k terminals.

However, more work is required. For set-cover, each element was either covered by the first-stage, or it was not; for cut problems, things are not so cut-and-dried, since both stages may help in severing a terminal from the root! So we divide S into two parts differently: the first part contains those nodes whose min-cut in G is large (since they belonged to S) but it fell by a constant factor in the graph $G \setminus \Phi^*$. These we call "low" nodes, and we use a Gomory-Hu tree based analysis to show that all low nodes can be completely separated from r by paying only $O(\Phi^*)$ more (this we show in Claim 4). The remaining "high" nodes continue to have a large min-cut in $G \setminus \Phi^*$, and for these we use the dual rounding idea sketched above to show a min-cut of $O(T^*)$ (this is proved in Claim 5). Together these claims imply Theorem 3.

To begin the proof of Theorem 3, let $H := G \setminus \Phi^*$, and let $S_h \subseteq S$ denote the "high" vertices whose min-cut from the root in H is at least $M := \frac{\beta}{2} \cdot \frac{T^*}{k}$. The following claim is essentially from Golovin et al. [10].

Claim 4 (Cutting Low Nodes). *If $T \geq T^*$, the minimum cut in H separating $S \setminus S_h$ from r costs at most $2 \cdot \Phi^*$.*

Claim 5 (Cutting High Nodes). *If $T \geq T^*$, the minimum r-S_h cut in H costs at most $\frac{\beta}{2} \cdot T^*$, when $\beta \geq \frac{10 \cdot e}{e-1}$.*

Proof: Consider a r-S_h max-flow in the graph $H = G \setminus \Phi^*$, and suppose it sends $\alpha_i \cdot M$ flow to vertex $i \in S_h$. By making copies of terminals, we can assume each $\alpha_i \in (0, 1]$; the k-robust min-cut problem remains unchanged under making copies. Hence if we show that $\sum_{i \in S_h} \alpha_i \leq k$, the total flow (which equals the min r-S_h cut) would be at most $k \cdot M = \frac{\beta}{2} \cdot T^*$, which would prove the claim. For a contradiction, we suppose that $\sum_{i \in S_h} \alpha_i > k$. We will now claim that there exists a subset $W \subseteq S_h$ with $|W| \leq k$ such that the min r-W cut is more than T^*, contradicting the fact that every k-set in H can be separated from r by a cut of value at most T^*. To find this set W, the following redistribution lemma (see [13] for proof) is useful.

Lemma 3 (Redistribution Lemma). *Let $N = (V, E)$ be a capacitated undirected graph. Let $X \subseteq V$ be a set of terminals such min-cut$_N(i, j) \geq 1$ for all nodes $i, j \in X$. For each $i \in X$, we are given a value $\epsilon_i \in (0, 1]$. Then for any*

integer $\ell \leq \sum_{i \in X} \epsilon_i$*, there exists a subset* $W \subseteq X$ *with* $|W| \leq \ell$ *vertices, and a feasible flow* f *in* N *from* X *to* W *so that (i) the total* f*-flow into* W *is at least* $\frac{1-e^{-1}}{4} \cdot \ell$ *and (ii) the* f*-flow out of* $i \in X$ *is at most* $\epsilon_i/4$.

We apply this lemma to $H = G \setminus \Phi^*$ with terminal set S_h, but with capacities scaled down by M. Since for any cut separating $x, y \in S_h$, the root r lies on one side on this cut (say on y's side), $\text{min-cut}_H(x, y) \geq M$—hence the scaled-down capacities satisfy the conditions of the lemma. Now set $\ell = k$, and $\epsilon_i := \alpha_i$ for each terminal $i \in S_h$; by the assumption $\sum_{i \in S_h} \epsilon_i = \sum_{i \in S_h} \alpha_i \geq k = \ell$. Hence Lemma 3 finds a subset $W \subseteq S_h$ with k vertices, and a flow f in (unscaled) graph H such that f sends a total of at least $\frac{1-1/e}{4} \cdot kM$ units into W, and at most $\frac{\alpha_i}{4} \cdot M$ units out of each $i \in S_h$. Also, there is a feasible flow g in the network H that simultaneously sends $\alpha_i \cdot M$ flow from the root to each $i \in S_h$, namely the max-flow from r to S_h. Hence the flow $\frac{g+4f}{5}$ is *feasible* in H, and sends $\frac{4}{5} \cdot \frac{1-1/e}{4} \cdot kM = \frac{1-1/e}{5} \cdot kM$ units from r into W. Finally, if $\beta > \frac{10 \cdot e}{e-1}$, we obtain that the min-cut in H separating W from r is greater than T^*: since $|W| \leq k$, this is a contradiction to the assumption that any set with at most k vertices can separated from the root in H at cost at most T^*. ■

From Claim 3 and Theorem 3, we obtain a $(3, \frac{\beta}{2}, \beta)$-discriminating algorithm for k-robust minimum cut, when $\beta \geq \frac{10e}{e-1}$. We set $\beta = \frac{10e}{e-1}$ and use Lemma 1 to infer that the approximation ratio of this algorithm is $\max\{3, \frac{\beta}{2\lambda} + \beta\} = \frac{\beta}{2\lambda} + \beta$. Since picking edges only in the second-stage is a trivial λ-approximation, the better of the two gives an approximation of $\min\{\frac{\beta}{2\lambda} + \beta,\ \lambda\} < 17$. Thus we have,

Theorem 4 (Min-cut Theorem). *There is a 17-approximation algorithm for* k*-robust minimum cut.*

5 Final Remarks

In this paper, we have presented a unified approach to directly solving k-robust covering problems and k-max-min problems. As mentioned in the introduction, one can show that solving the k-max-min problem also leads to a k-robust algorithm—we give a general reduction in [13]. While this general reduction leads to poorer approximation guarantees for the cardinality case, it easily extends to more general cases. Indeed, if the uncertainty sets for robust problems are not just defined by cardinality constraints, we can ask: which families of downwards-closed sets can we devise robust algorithms for? In [13] we show how to incorporate intersections of matroid-type and knapsack-type uncertainty sets.

Our work suggests several general directions for research. While the results for the k-robust case are fairly tight, can we improve on our results for general uncertainty sets to match those for the cardinality case? Can we devise algorithms to handle one matroid constraint that are as simple as our algorithms for the cardinality case? An intriguing specific problem is to find a constant factor approximation for the robust Steiner forest problem with explicit scenarios.

Acknowledgments. We thank Chandra Chekuri, Ravishankar Krishnaswamy, Danny Segev, and Maxim Sviridenko for invaluable discussions.

References

1. Agrawal, S., Ding, Y., Saberi, A., Ye, Y.: Correlation Robust Stochastic Optimization. In: SODA (2010)
2. Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., Naor, S.: The Online Set Cover Problem. In: STOC, pp. 100–105 (2003)
3. Calinescu, G., Chekuri, C., Pál, M., Vondrák, J.: Maximizing a monotone submodular function under a matroid constraint. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 182–196. Springer, Heidelberg (2007)
4. Dhamdhere, K., Goyal, V., Ravi, R., Singh, M.: How to pay, come what may: Approximation algorithms for demand-robust covering problems. In: FOCS, pp. 367–378 (2005)
5. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. J. Comput. System Sci. 69(3), 485–497 (2004)
6. Feige, U., Jain, K., Mahdian, M., Mirrokni, V.S.: Robust combinatorial optimization with exponential scenarios. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 439–453. Springer, Heidelberg (2007)
7. Fisher, M.L., Nemhauser, G.L., Wolsey, L.A.: An analysis of approximations for maximizing submodular set functions II. Mathematical Programming Study 8, 73–87 (1978)
8. Gandhi, R., Khuller, S., Srinivasan, A.: Approximation algorithms for partial covering problems. J. Algorithms 53(1), 55–84 (2004)
9. Garg, N.: Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In: STOC 2005: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, pp. 396–402 (2005)
10. Golovin, D., Goyal, V., Ravi, R.: Pay today for a rainy day: improved approximation algorithms for demand-robust min-cut and shortest path problems. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 206–217. Springer, Heidelberg (2006)
11. Golovin, D., Nagarajan, V., Singh, M.: Approximating the k-multicut problem. In: SODA 2006: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, pp. 621–630 (2006)
12. Gupta, A., Hajiaghayi, M.T., Nagarajan, V., Ravi, R.: Dial a ride from k-forest. In: Proceedings of the 15th Annual European Symposium on Algorithms, pp. 241–252 (2007)
13. Gupta, A., Nagarajan, V., Ravi, R.: Thresholded covering algorithms for robust and max-min optimization (2009), (full version) http://arxiv.org/abs/0912.1045
14. Immorlica, N., Karger, D., Minkoff, M., Mirrokni, V.S.: On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In: SODA, pp. 691–700 (2004)
15. Khandekar, R., Kortsarz, G., Mirrokni, V.S., Salavatipour, M.R.: Two-stage robust network design with exponential scenarios. In: Halperin, D., Mehlhorn, K. (eds.) Esa 2008. LNCS, vol. 5193, pp. 589–600. Springer, Heidelberg (2008)
16. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)

17. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions I. Mathematical Programming 14, 265–294 (1978)
18. Räcke, H.: Optimal hierarchical decompositions for congestion minimization in networks. In: STOC, pp. 255–264 (2008)
19. Ravi, R., Sinha, A.: Hedging uncertainty: approximation algorithms for stochastic optimization problems. In: Bienstock, D., Nemhauser, G.L. (eds.) IPCO 2004. LNCS, vol. 3064, pp. 101–115. Springer, Heidelberg (2004)
20. Shmoys, D., Swamy, C.: Stochastic Optimization is (almost) as Easy as Deterministic Optimization. In: FOCS, pp. 228–237 (2004)
21. Slavík, P.: Improved performance of the greedy algorithm for partial cover. Inf. Process. Lett. 64(5), 251–254 (1997)
22. Srinivasan, A.: Improved approximation guarantees for packing and covering integer programs. SIAM J. Comput. 29(2), 648–670 (1999)
23. Sviridenko, M.: A note on maximizing a submodular set function subject to knapsack constraint. Operations Research Letters 32, 41–43 (2004)
24. Swamy, C.: Algorithms for Probabilistically-Constrained Models of Risk-Averse Stochastic Optimization with Black-Box Distributions (2008), http://arxiv.org/abs/0805.0389
25. Vondrák, J.: Optimal approximation for the submodular welfare problem in the value oracle model. In: STOC, pp. 67–74 (2008)

Graph Homomorphisms with Complex Values: A Dichotomy Theorem
(Extended Abstract)*

Jin-Yi Cai[1,**], Xi Chen[2,***], and Pinyan Lu[3]

[1] University of Wisconsin-Madison
jyc@cs.wisc.edu
[2] University of Southern California
csxichen@gmail.com
[3] Microsoft Research Asia
pinyanl@microsoft.com

Abstract. Graph homomorphism problem has been studied intensively. Given an $m \times m$ symmetric matrix $\mathbf{A}$, the graph homomorphism function $Z_{\mathbf{A}}(G)$ is defined as

$$Z_{\mathbf{A}}(G) = \sum_{\xi: V \to [m]} \prod_{(u,v) \in E} A_{\xi(u),\xi(v)},$$

where $G = (V, E)$ is any undirected graph. The function $Z_{\mathbf{A}}(G)$ can encode many interesting graph properties, including counting vertex covers and k-colorings. We study the computational complexity of $Z_{\mathbf{A}}(G)$, for arbitrary complex valued symmetric matrices $\mathbf{A}$. Building on the work by Dyer and Greenhill [1], Bulatov and Grohe [2], and especially the recent beautiful work by Goldberg, Grohe, Jerrum and Thurley [3], we prove a complete dichotomy theorem for this problem.

1 Introduction

Graph homomorphism has been studied intensively over the years [4,5,1,6,2,7,3]. Given two graphs G and H, a graph homomorphism from G to H is a map $f : V(G) \to V(H)$ such that, whenever (u, v) is an edge in G, $(f(u), f(v))$ is an edge in H. The counting problem for graph homomorphism is then to compute the number of homomorphisms from G to H. For a fixed graph H this problem is also known as the $\#H$-coloring problem. In 1967, Lovász [4] proved that H and H' are isomorphic if and only if for all G, the number of homomorphisms from G to H and from G to H' are the same. Graph homomorphisms and the

* The graph homomorphism function is also known as the partition function.

** Supported by NSF CCF-0830488 and CCF-0914969.

*** Most of the work done while the author was a postdoc at the Institute for Advanced Study and Princeton University. Supported by Grants CCF-0832797, DMS-0635607, and the USC Viterbi School of Engineering Startup Fund (from Shang-Hua Teng).

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 275–286, 2010.

associated partition functions defined next in (1) provide an extremely elegant and general notion of *graph properties* [5].

In this paper, all the graphs considered are undirected. We follow the standard definitions: G is allowed to have multiple edges; H can have loops, multiple edges and more generally, edge weights. (The standard definition of graph homomorphism does not allow self-loops for G. However, our result is stronger: We prove polynomial-time tractability even for graphs G with self-loops; and at the same time, our hardness results hold for the more restricted case of G without self-loops.) Formally, let $\mathbf{A} = (A_{i,j})$ be an $m \times m$ symmetric matrix. Then for any undirected graph $G = (V, E)$, we define

$$Z_{\mathbf{A}}(G) = \sum_{\xi: V \to [m]} \mathrm{wt}_{\mathbf{A}}(G, \xi), \tag{1}$$

where

$$\mathrm{wt}_{\mathbf{A}}(G, \xi) = \prod_{(u,v) \in E} A_{\xi(u), \xi(v)}$$

is called the *weight* of ξ. $Z_{\mathbf{A}}(G)$ is also called the *partition function* from statistical physics.

Graph homomorphisms can express many natural graph properties. For example, if we take H to be the graph over two vertices $\{0, 1\}$, with an edge $(0, 1)$ and a loop at 1, then a graph homomorphism from G to H corresponds to a Vertex Cover of G, and the counting problem simply counts the number of vertex covers. As another example, if H is the complete graph over k vertices without self-loops, then the problem is exactly the k-Coloring problem for G. Many additional graph invariants can be expressed as $Z_{\mathbf{A}}(G)$ for appropriate $\mathbf{A}$. Consider the 2×2 Hadamard matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{2}$$

where we index the rows and columns by $\{0, 1\}$. In $Z_{\mathbf{H}}(G)$, $\mathrm{wt}_{\mathbf{H}}(G, \xi) = \pm 1$ and is -1 precisely when the induced subgraph of G on $\xi^{-1}(1)$ has an odd number of edges. Therefore, $(2^n - Z_{\mathbf{H}}(G))/2$ is the *number* of induced subgraphs of G with an odd number of edges. In [6], Freedman, Lovász, and Schrijver characterized what graph functions can be expressed as $Z_{\mathbf{A}}(G)$.

In this paper, we study the computational complexity of $Z_{\mathbf{A}}(G)$ where $\mathbf{A}$ is an arbitrary fixed $m \times m$ symmetric matrix over the *complex numbers*, and G is an input graph. The complexity question of $Z_{\mathbf{A}}(G)$ has also been intensively studied. In [8] Hell and Nešetřil first studied the complexity of the decision problem for H-coloring (that is, given an undirected graph G, decide whether there exists a graph homomorphism from G to H or not). They proved that for any undirected H represented by a symmetric $\{0, 1\}$-matrix, H-coloring is either in P or NP-complete. Dyer and Greenhill [1] then studied the counting version and proved that for any $\{0, 1\}$ symmetric matrix $\mathbf{A}$, computing $Z_{\mathbf{A}}(G)$ is either in P or #P-hard. Bulatov and Grohe [2] then gave a sweeping generalization of this theorem to all non-negative symmetric matrices (see Theorem 2 for the precise

statement). In a beautiful paper of exceptional depth and vision, Goldberg, Jerrum, Grohe, and Thurley [3] proved a complexity dichotomy theorem for all real valued symmetric matrices $\mathbf{A}$. Recently, Thurley [9] has given a dichotomy theorem for complex Hermitian matrices. The polynomial-time tractability result of the present paper [10] is used in [9]. Results of this type are called *complexity dichotomy* theorems: They state, for a wide class of problems, every member of the class is either tractable (i.e., solvable in P) or intractable (in this case, hard for the counting class #P). The first well-known dichotomy theorem is Schaefer's theorem [11], and many more are proved in the study on constraint satisfaction problems (CSP in short) [12]. In particular, the famous dichotomy conjecture by Feder and Vardi [13] on Decision CSP motivated much of subsequent work.

The result of Bulatov and Grohe [2] is both sweeping and enormously applicable. It completely solves the problem for non-negative symmetric matrices $\mathbf{A}$. However, when we are dealing with non-negative numbers, there are no cancellations in the exponential sum $Z_{\mathbf{A}}(G)$ defined by (1). But these potential cancellations, when $\mathbf{A}$ is either a real or a complex matrix, may in fact be the source of surprisingly efficient algorithms for $Z_{\mathbf{A}}(G)$. The occurrence of these cancellations, or the mere possibility of such occurrence, makes proving any complexity dichotomies more difficult. Such a proof must identify all polynomial-time algorithms utilizing the potential cancellations, such as those found in holographic algorithms [14,15,16], and at the same time carve out exactly what's left. This situation is similar to *monotone* versus *non-monotone* circuit complexity. It turns out that indeed there are more interesting tractable (i.e., computable in polynomial time) cases over the reals without the restriction of non-negativity. The 2×2 Hadamard matrix $\mathbf{H}$ turns out to be one of such cases, as was shown by Goldberg, Jerrum, Grohe and Thurley in [3].

While positive and negative real numbers provide the possibility of cancellations, over the complex domain there is a significantly richer variety of possible cancellations. We independently came to the tractability of $Z_{\mathbf{H}}(\cdot)$ from a slightly different angle. In [17] we were studying a certain type of constraint satisfaction problems. This is motivated by the investigations of a class of counting problems called *Holant problems*, and it is connected with the technique called holographic reductions introduced by Valiant [14,15]. Let us briefly describe this framework.

A *signature grid* $\Omega = (G, \mathcal{F})$ is a tuple, where $G = (V, E)$ is an undirected graph, and each $v \in V$ is attached a function $F_v \in \mathcal{F}$. An edge assignment σ for every $e \in E$ gives an evaluation

$$\prod_{v \in V} F_v\big(\sigma\,|_{E(v)}\big),$$

where $E(v)$ denotes the incident edges of v. The counting problem on an input instance Ω is to compute

$$\text{Holant}(\Omega) = \sum_{\text{edge assignments } \sigma} \prod_{v \in V} F_v\big(\sigma\,|_{E(v)}\big).$$

For example, if we take $\sigma : E \to \{0, 1\}$ and attach the Exact-One function at every $v \in V$, then Holant(Ω) computes the number of perfect matchings of G. Incidentally, Freedman, Lovász, and Schrijver [6] showed that counting perfect

matchings *cannot* be expressed as $Z_{\mathbf{A}}(\cdot)$ for any real $\mathbf{A}$. However, every function $Z_{\mathbf{A}}(G)$ (vertex assignment) *can* be simulated by Holant(Ω) (edge assignment).

We discovered that the following three families of functions

$$\mathcal{F}_1 = \left\{ \lambda([1,0]^{\otimes k} + i^r[0,\ \ 1]^{\otimes k}) \;\middle|\; \lambda \in \mathbb{C},\ k = 1,2,\ldots,\ \text{and } r = 0,1,2,3 \right\};$$
$$\mathcal{F}_2 = \left\{ \lambda([1,1]^{\otimes k} + i^r[1,-1]^{\otimes k}) \;\middle|\; \lambda \in \mathbb{C},\ k = 1,2,\ldots,\ \text{and } r = 0,1,2,3 \right\};$$
$$\mathcal{F}_3 = \left\{ \lambda([1,\ i]^{\otimes k} + i^r[1,\ -i]^{\otimes k}) \;\middle|\; \lambda \in \mathbb{C},\ k = 1,2,\ldots,\ \text{and } r = 0,1,2,3 \right\}$$

give rise to tractable problems: Holant(Ω), for any $\Omega = (G, \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3)$, can be computed in polynomial time (we listed functions in $\mathcal{F}_i$ in the form of truth tables on k boolean variables).

Note that by taking $r = 1, k = 2$ and $\lambda = (1+i)^{-1}$ in $\mathcal{F}_3$, we recover exactly the binary function which corresponds to the 2×2 Hadamard matrix $\mathbf{H}$ in (2); By taking $r = 0$ and $\lambda = 1$ in $\mathcal{F}_1$, we get the EQUALITY function over k bits for $k \geq 1$. Combining these two facts, one can show easily that $Z_{\mathbf{H}}(G)$ is a special case of Holant(Ω), and the tractability of Holant(Ω) implies that of $Z_{\mathbf{H}}(\cdot)$.

However, more instructive for us is the natural way in which complex numbers appeared in such counting problems, especially when applying holographic reductions. One can say that the presence of powers of $i = \sqrt{-1}$ in $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ "reveals" the *true* nature of $\mathbf{H}$ as belonging to a family of *tractable* counting problems where complex numbers are the right language. Thus the investigation of the complexity of $Z_{\mathbf{A}}(\cdot)$ for symmetric complex matrices not only is a natural generalization, but also can reveal the inner unity and its deeper structural properties. Interested readers can find more details in [17].

Our investigation of complex-valued graph homomorphisms is also motivated by partition functions in quantum physics. In classical statistical physics, partition functions are always real-valued. However, in a generic quantum system for which complex numbers are the correct language, the partition function is then complex-valued in general [18]. In particular, if the physics model is over a discrete graph and is non-orientable, then the edge weights are given by a symmetric complex matrix.

Our main result is the following complexity dichotomy theorem:

Theorem 1. *Let* $\mathbf{A}$ *be a symmetric complex matrix, then* $Z_{\mathbf{A}}(\cdot)$ *is either* in P *or* #P-hard.

Due to the complexity of the proof of Theorem 1, both in terms of its overall proof structure and in terms of the technical difficulty, we will only give a high level description of the proof in this extended abstract. The full version of the paper can be found in [10]. We remark that for algebraic $\mathbf{A}$, we can show that the dichotomy is decidable, but this will be discussed in a future publication.

2 Preliminaries

For a positive integer n, we let $[n]$ denote the set $\{1, 2, \ldots, n\}$. We also let $[m : n]$ where $m \leq n$, denote the set $\{m, \ldots, n\}$. Given $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$, we use

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i \cdot \overline{y_i}$$

to denote their inner product and use $\mathbf{x} \circ \mathbf{y}$ to denote their Hadamard product: $\mathbf{z} = \mathbf{x} \circ \mathbf{y} \in \mathbb{C}^n$, where

$$z_i = x_i \cdot y_i, \quad \text{for all } i \in [n].$$

For a positive integer N, we let ω_N denote $e^{2\pi i/N}$, a primitive N*th* root of unity.

Let $\mathbf{A} = (A_{i,j})$ be a $k \times \ell$ matrix, and let $\mathbf{B} = (B_{i,j})$ be an $m \times n$ matrix. We use $\mathbf{A}_{i,*}$ to denote the ith row, and $\mathbf{A}_{*,j}$ to denote the jth column of $\mathbf{A}$. We let $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$ denote the tensor product of $\mathbf{A}$ and $\mathbf{B}$: $\mathbf{C}$ is a $km \times \ell n$ matrix whose rows and columns are indexed by $[k] \times [m]$ and $[\ell] \times [n]$ such that

$$C_{(i_1,i_2),(j_1,j_2)} = A_{i_1,j_1} \cdot B_{i_2,j_2}, \quad \text{for all } i_1 \in [k],\ i_2 \in [m],\ j_1 \in [\ell],\ j_2 \in [n].$$

Given any symmetric $n \times n$ matrix $\mathbf{A}$ we build an undirected graph $G = (V, E)$ as follows: $V = [n]$ and $ij \in E$ if and only if $A_{i,j} \neq 0$. We say $\mathbf{A}$ is *connected* if G is connected; and we say $\mathbf{A}$ has connected components $\mathbf{A}_1, \ldots, \mathbf{A}_s$, if the connected components of G are $V_1, \ldots, V_s$ and for every i, $\mathbf{A}_i$ is the $|V_i| \times |V_i|$ sub-matrix of $\mathbf{A}$ restricted by $V_i \subseteq [n]$. Moreover, we say $\mathbf{A}$ is *bipartite*, if G is bipartite; otherwise $\mathbf{A}$ is *non-bipartite*. Finally we say $\mathbf{C}$ is the *bipartisation* of a matrix $\mathbf{F}$ if

$$\mathbf{C} = \begin{pmatrix} \mathbf{0} & \mathbf{F} \\ \mathbf{F}^T & \mathbf{0} \end{pmatrix}.$$

Note that $\mathbf{C}$ is always a symmetric matrix no matter whether $\mathbf{F}$ is or is not.

Now let $\mathbf{A}$ be any symmetric complex matrix. It defines a problem $\mathsf{EVAL}(\mathbf{A})$ as follows: Given an undirected graph G, compute $Z_{\mathbf{A}}(G)$. To study the complexity of $\mathsf{EVAL}(\mathbf{A})$ and prove Theorem 1, we need to introduce a larger class of EVAL problems. We will define a generalized partition function which not only has edge weights but also has vertex weights. Moreover, the vertex weights also depend on the degrees of vertices of G, modulo some integer modulus. Formally, let $\mathbf{C} \in \mathbb{C}^{m \times m}$ be a symmetric matrix and

$$\mathfrak{D} = \left\{ \mathbf{D}^{[0]}, \mathbf{D}^{[1]}, \ldots, \mathbf{D}^{[N-1]} \right\}$$

be a sequence of N diagonal matrices in $\mathbb{C}^{m \times m}$ for some $N \geq 1$. We use $D_i^{[r]}$ to denote the $(i,i)th$ diagonal entry of $\mathbf{D}^{[r]}$. We define the problem $\mathsf{EVAL}(\mathbf{C}, \mathfrak{D})$ as follows: Given $G = (V, E)$, compute

$$Z_{\mathbf{C},\mathfrak{D}}(G) = \sum_{\xi : V \to [m]} \mathrm{wt}_{\mathbf{C},\mathfrak{D}}(G, \xi),$$

where

$$\mathrm{wt}_{\mathbf{C},\mathfrak{D}}(G, \xi) = \left(\prod_{(u,v) \in E} C_{\xi(u),\xi(v)} \right) \left(\prod_{v \in V} D_{\xi(v)}^{[\deg(v) \bmod N]} \right)$$

and $\deg(v)$ denotes the degree of v in G.

We also formally state the dichotomy of Bulatov and Grohe [2] as follows:

Theorem 2 ([2]). *Let* $\mathbf{A}$ *be an* $m \times m$ *symmetric, connected, and non-negative matrix, then*

1. *If* $\mathbf{A}$ *is bipartite and* $m \geq 2$*, then* $\mathsf{EVAL}(\mathbf{A})$ *is* in P *if* $\operatorname{rank}(\mathbf{A}) = 2$*, and is* #P-hard *otherwise;*
2. *If* $\mathbf{A}$ *is non-bipartite, then* $\mathsf{EVAL}(\mathbf{A})$ *is* in P *if* $\operatorname{rank}(\mathbf{A}) \leq 1$*, and is* #P-hard *otherwise.*

One technical issue is the model of computation with complex numbers. We can take any reasonable model of real or complex number computation as long as arithmetic operations such as $+$ and $\times$ are computable, and equality is decidable [19,20]. For the most part, this issue of computation model seems not central to this paper, in part because we always consider the symmetric complex matrix $\mathbf{A}$ to be fixed and the complexity measure is on the size of the input G. In the most restrictive sense, we can require entries of $\mathbf{A}$ to be algebraic numbers, and the theorem holds in the strict bit model of Turing machines. Over the algebraic numbers, we can prove that our dichotomy theorem gives a decidable criterion. This will be discussed in a future publication.

3 A High Level Description of the Proof

The first step, in the proof of Theorem 1, is to reduce the problem to connected graphs and matrices. Let $\mathbf{A}$ be an $m \times m$ symmetric complex matrix. It is easy to prove that, if the input graph G has connected components G_i, then

$$Z_{\mathbf{A}}(G) = \prod_i Z_{\mathbf{A}}(G_i);$$

and if G is connected, and $\mathbf{A}$ has connected components $\mathbf{A}_j$, then

$$Z_{\mathbf{A}}(G) = \sum_j Z_{\mathbf{A}_j}(G).$$

Therefore, if every $Z_{\mathbf{A}_j}(\cdot)$ is computable in P, then so is $Z_{\mathbf{A}}(\cdot)$.

The hardness direction is less obvious. Suppose $Z_{\mathbf{A}_j}(\cdot)$ is #P-hard for some j. We want to prove that $Z_{\mathbf{A}}(\cdot)$ is also #P-hard. We prove this by showing that computing $Z_{\mathbf{A}_j}(\cdot)$ is reducible to computing $Z_{\mathbf{A}}(\cdot)$. Let G be an arbitrary input graph. To compute $Z_{\mathbf{A}_j}(G)$, it suffices to compute $Z_{\mathbf{A}_j}(G_i)$ for every connected component G_i of G. Therefore, we may just assume that G is connected. Define a *pinning* version of $Z_{\mathbf{A}}(\cdot)$ as follows: For any chosen vertex $w \in V(G)$ and any $k \in [m]$, we let

$$Z_{\mathbf{A}}(G, w, k) = \sum_{\xi : V \to [m],\, \xi(w) = k} \operatorname{wt}_{\mathbf{A}}(G, \xi).$$

Then we prove a *Pinning* Lemma (Lemma 4.1 in the full version), which states that the problem of computing $Z_{\mathbf{A}}(\cdot)$ is polynomial-time equivalent to the problem of computing $Z_{\mathbf{A}}(\cdot, \cdot, \cdot)$. Note that if V_j denotes the subset of $[m]$ such that $\mathbf{A}_j$ is the sub-matrix of $\mathbf{A}$ restricted by V_j, then for a connected G, we have

$$Z_{\mathbf{A}_j}(G) = \sum_{k \in V_j} Z_{\mathbf{A}}(G, w, k),$$

which gives us a polynomial-time reduction from $Z_{\mathbf{A}_j}(\cdot)$ to $Z_{\mathbf{A}}(\cdot)$.

It turns out that we actually need a total of three *Pinning Lemmas* (Lemma 4.1, 4.2 and 8.2 in the full version [10]). The proof of the first Pinning Lemma is an adaptation (to the complex numbers) of a similar lemma proved in [3]. For the other two Pinning Lemmas, the proofs are a bit more involved. We remark that all our Pinning Lemmas only show the *existence* of polynomial-time reductions between $Z_{\mathbf{A}}(\cdot)$ and $Z_{\mathbf{A}}(\cdot,\cdot,\cdot)$, but they do not *constructively* produce such reductions, given $\mathbf{A}$. We also remark that the proof of the Pinning Lemma in [3] uses a recent result [21] by Lovász for real matrices. This result is not known for complex matrices. We give a direct proof of our three lemmas without using the result of Lovász.

After this preliminary step we restrict to *connected* and *symmetric* $\mathbf{A}$. As indicated, to our work the two most influential predecessor papers are by Bulatov and Grohe [2], and by Goldberg et. al. [3]. In both papers, the polynomial-time algorithms for those tractable cases are relatively straightforward. The difficult part of the proof is to prove that, in all other cases, the problem of computing $Z_{\mathbf{A}}(\cdot)$ is #P-hard. Over the complex numbers, new difficulties arise in both the tractability and the hardness part of the proof:

- For the tractability part, it turns out that *Gauss sums* play a crucial role, and we devise new polynomial-time algorithms;
- For the hardness part, the difficulty already starts with the most basic technique called *gadget constructions*, to be discussed shortly. Technically, all our hardness proofs are done by reductions to the non-negative case using Bulatov-Grohe [2]. The difficulty with proving hardness for complex matrices goes deeper than appearance.

In [2] and [3] the polynomial-time algorithms for the tractable cases are previously known algorithms. However, the complex numbers afford a much richer variety of cancellations, which could lead to surprisingly efficient algorithms for computing $Z_{\mathbf{A}}(\cdot)$ for certain complex matrices $\mathbf{A}$. It turns out that this is indeed the case, and we obtain additional non-trivial tractable cases. These boil down to the following class of problems:

> $Z_q(\cdot)$: Let $q = p^k$ be a (fixed) prime power for some prime p and positive integer k. The input of $Z_q(\cdot)$ is a quadratic polynomial
>
> $$f(x_1, x_2, \ldots, x_n) = \textstyle\sum_{i,j\in[n]} a_{i,j} x_i x_j, \quad \text{where } a_{i,j} \in \mathbb{Z}_q \text{ for all } i, j;$$
>
> and the output is
>
> $$Z_q(f) = \sum_{x_1,\ldots,x_n \in \mathbb{Z}_q} \omega_q^{f(x_1,\ldots,x_n)}.$$

We show that for any fixed prime power q, $Z_q(\cdot)$ can be computed in polynomial time. The tractability part of our dichotomy theorem is then done by reducing $Z_{\mathbf{A}}(\cdot)$, with $\mathbf{A}$ satisfying certain structural properties (which we will describe in the rest of this section), to $Z_q(\cdot)$ for some appropriate prime power q. While the

corresponding sums for finite fields are known to be computable in polynomial time [22], in particular this includes the special case of $\mathbb{Z}_2$ which was used in [3], our algorithm over rings $\mathbb{Z}_q$ is new and should be of independent interest. The algorithm is presented in Section 12 of the full version. (The theorem of [3] can be considered as the case where the only root of unity is -1. However, we note that odd prime powers generally behave differently than powers of 2.)

We now briefly describe the proof structure of the dichotomy theorem. First let $\mathbf{A}$ be a connected and symmetric complex matrix. As has already been used many times before, a key tool in the hardness part is the use of *graph gadgets*. With a graph gadget, we can take any input graph G, and produce a modified graph G^* with the following property: one can transform the fixed matrix $\mathbf{A}$ to a suitably modified matrix $\mathbf{A}^*$ (which only depends on $\mathbf{A}$ and the graph gadget and does not depend on the input graph G) such that

$$Z_{\mathbf{A}^*}(G) = Z_{\mathbf{A}}(G^*), \quad \text{for any undirected graph } G.$$

This gives us a polynomial-time reduction from $Z_{\mathbf{A}^*}(\cdot)$ to $Z_{\mathbf{A}}(\cdot)$.

A simple example of this maneuver is called *thickening*: Given any input G one replaces each edge of G by t parallel edges to get G^*. It is easy to see that if $\mathbf{A}^*$ is obtained from $\mathbf{A}$ by replacing each entry $A_{i,j}$ with its t^{th} power $(A_{i,j})^t$, then the equation above holds and we get a reduction from $Z_{\mathbf{A}^*}(\cdot)$ to $Z_{\mathbf{A}}(\cdot)$. In particular, if $\mathbf{A}$ is a real matrix (as in the case of [3]) and t is even, this always produces a non-negative $\mathbf{A}^*$ to which one may apply the Bulatov-Grohe result:

- either $Z_{\mathbf{A}}(\cdot)$ is #P-hard, in which case we are already done;
- or $Z_{\mathbf{A}}(\cdot)$ is not #P-hard, in which case $Z_{\mathbf{A}^*}(\cdot)$ cannot be #P-hard (due to the reduction from $Z_{\mathbf{A}^*}(\cdot)$ to $Z_{\mathbf{A}}(\cdot)$) and $\mathbf{A}^*$, derived from $\mathbf{A}$, must pass the Bulatov-Grohe tractability test as described in Theorem 2.

As a result, if we assume $Z_{\mathbf{A}}(\cdot)$ is not #P-hard, then the matrix $\mathbf{A}$ must satisfy certain necessary structural conditions. The big picture[1] of the proof of the dichotomy theorem is then to design various *graph gadgets* to show that, assuming $Z_{\mathbf{A}}(\cdot)$ is not #P-hard, $\mathbf{A}$ must satisfy a *collection* of strong necessary conditions over its complex entries $A_{i,j}$. To finish the proof, we show that for every $\mathbf{A}$ that satisfies all these conditions, one can reduce $Z_{\mathbf{A}}(\cdot)$ to $Z_q(\cdot)$ for some appropriate prime power q (which only depends on $\mathbf{A}$) and thus, $Z_{\mathbf{A}}(\cdot)$ is tractable.

For complex matrices $\mathbf{A}$ we immediately encountered the following difficulty. Any graph gadget can only produce a new matrix $\mathbf{A}^*$ whose entries are derived from those of $\mathbf{A}$ by arithmetic operations $+$ and $\times$. While for real numbers any even power guarantees a non-negative quantity, no obvious operations over the complex numbers have this property. Pointedly, *conjugation* is *not* an arithmetic operation. However, it is also clear that for *roots of unity* one *can* easily produce conjugation by multiplications.

[1] The exact proof structure, however, is different from this very high-level description, which will become clear through the rest of the section.

Thus, the proof of our dichotomy starts with a process to replace an arbitrary complex matrix $\mathbf{A}$ with a *purified* complex matrix which has a very special form. It turns out that we must separate out the cases where $\mathbf{A}$ is bipartite or non-bipartite. A purified bipartite (and symmetric/connected) matrix takes the following form: It is the bipartisation of a matrix $\mathbf{B}$, where (note that $\mathbf{B}$ is not necessarily symmetric here)

$$\mathbf{B} = \begin{pmatrix} \mu_1 & & & \\ & \mu_2 & & \\ & & \ddots & \\ & & & \mu_k \end{pmatrix} \begin{pmatrix} \zeta_{1,1} & \zeta_{1,2} & \cdots & \zeta_{1,m-k} \\ \zeta_{2,1} & \zeta_{2,2} & \cdots & \zeta_{2,m-k} \\ \vdots & \vdots & \ddots & \vdots \\ \zeta_{k,1} & \zeta_{k,2} & \cdots & \zeta_{k,m-k} \end{pmatrix} \begin{pmatrix} \mu_{k+1} & & & \\ & \mu_{k+2} & & \\ & & \ddots & \\ & & & \mu_m \end{pmatrix},$$

for some $1 \leq k < m$, in which every $\mu_i > 0$ and every $\zeta_{i,j}$ is a root of unity.

The claim (Theorem 5.1 of the full version) is that, for any symmetric, connected and bipartite complex $m \times m$ matrix $\mathbf{A}$, either we can already show the #P-hardness of $Z_{\mathbf{A}}(\cdot)$; or there exists an $m \times m$ symmetric, connected and *purified* bipartite matrix $\mathbf{A}'$ such that computing $Z_{\mathbf{A}'}(\cdot)$ is polynomial-time equivalent to computing $Z_{\mathbf{A}}(\cdot)$. For non-bipartite $\mathbf{A}$, a corresponding statement holds (see Theorem 6.1 of the full version). For convenience, in the discussion below, we only focus on the bipartite case.

Continuing now with a purified and bipartite matrix $\mathbf{A}'$, the next step is to further regularize its entries. In particular, we need to combine those rows and columns of $\mathbf{A}'$ where they are essentially the same apart from a multiple of a root of unity. This process is called *Cyclotomic Reduction*. In order to carry out this process, we need to use the more general counting problems $\mathsf{EVAL}(\mathbf{C}, \mathfrak{D})$, as defined earlier in Section 2. We also need to introduce the following special matrices called *discrete unitary* matrices:

Definition 1 (Discrete Unitary Matrices). *Let $\mathbf{F} \in \mathbb{C}^{m \times m}$ be a matrix with entries $(F_{i,j})$. We say $\mathbf{F}$ is an M-*discrete unitary matrix *for some positive integer M, if it satisfies the following conditions:*

1. *Every entry $F_{i,j}$ of $\mathbf{F}$ is a root of unity, and*
$$M = \operatorname{lcm}\left\{ \text{the order of } F_{i,j} : i, j \in [m] \right\};$$
2. *$F_{1,i} = F_{i,1} = 1$ for all $i \in [m]$, and for all $i \neq j \in [m]$,*
$$\langle \mathbf{F}_{i,*}, \mathbf{F}_{j,*} \rangle = 0 \quad \text{and} \quad \langle \mathbf{F}_{*,i}, \mathbf{F}_{*,j} \rangle = 0.$$

Given any purified bipartite matrix $\mathbf{A}'$, we continue to show that: $Z_{\mathbf{A}'}(\cdot)$ is either #P-hard or polynomial-time equivalent to $Z_{\mathbf{C},\mathfrak{D}}(\cdot)$ for some $\mathbf{C} \in \mathbb{C}^{2n \times 2n}$ and some $\mathfrak{D}$ of diagonal matrices from $\mathbb{C}^{2n \times 2n}$ where $\mathbf{C}$ is the bipartisation of a discrete unitary matrix, for some positive integer n. In addition to requiring $\mathbf{C}$ to be the bipartisation of a discrete unitary matrix, there are further stringent requirements for $\mathfrak{D}$; otherwise $Z_{\mathbf{A}'}(\cdot)$ is #P-hard. The detailed statement can be found in Theorem 5.2 and 5.3 of the full version, summarized in properties $(\mathcal{U}_1)$ to $(\mathcal{U}_5)$. Roughly speaking, these requirements are:

– The first diagonal matrix $\mathbf{D}^{[0]}$ in $\mathfrak{D}$ must be the identity matrix; and
– For every diagonal matrix $\mathbf{D}^{[r]}$ in $\mathfrak{D}$, any of its entries is either zero or a root of unity.

We call these requirements over $\mathbf{C}$ and $\mathfrak{D}$, with some abuse of terminology, the *discrete unitary requirements*. The proof of these requirements is very involved and among the most difficult in the paper.

Next, assume that we have a problem $(\mathbf{C}, \mathfrak{D})$, satisfying the discrete unitary requirements and in particular, $\mathbf{C}$ is the bipartisation of $\mathbf{F}$.

Definition 2. *Let $q > 1$ be a prime power. Then the following $q \times q$ matrix $\boldsymbol{\mathcal{F}}_q$ is called the* q-Fourier matrix*: the (x, y)th entry of $\boldsymbol{\mathcal{F}}_q$, $x, y \in [0 : q-1]$, is ω_q^{xy}.*

We show that, either computing $Z_{\mathbf{C},\mathfrak{D}}(\cdot)$ is #P-hard; or after permuting the rows/columns $\mathbf{F}$ becomes the *tensor product* of a collection of Fourier matrices:

$$\boldsymbol{\mathcal{F}}_{q_1} \otimes \boldsymbol{\mathcal{F}}_{q_2} \otimes \cdots \otimes \boldsymbol{\mathcal{F}}_{q_d}, \quad \text{where } d \geq 1 \text{ and every } q_i \text{ is a prime power.}$$

Basically we show that even with the stringent conditions put on the pair $(\mathbf{C}, \mathfrak{D})$ by the discrete unitary requirements, most of them will still be #P-hard, unless $\mathbf{F}$ has the further property of being a tensor product of Fourier matrices. Being able to express it as a tensor product of Fourier matrices finally brings in group theory and Gauss sums. It gives us a canonical way of writing the entries of $\mathbf{F}$ in a closed form (Here we assume that an appropriate permutation of rows and columns has already been applied to $\mathbf{F}$, as well as $\mathbf{C}$ and $\mathfrak{D}$). More exactly, we can index the rows and columns of $\mathbf{F}$ using

$$\mathbf{x} = (x_1, x_2, \ldots, x_d) \text{ and } \mathbf{y} = (y_1, y_2, \ldots, y_d) \in \mathbb{Z}_{q_1} \times \mathbb{Z}_{q_2} \times \cdots \times \mathbb{Z}_{q_d},$$

respectively, such that

$$F_{\mathbf{x},\mathbf{y}} = \prod_{i \in [d]} \omega_{q_i}^{x_i y_i}, \quad \text{for all } \mathbf{x} \text{ and } \mathbf{y}.$$

Assume $q_1, \ldots, q_d$ are powers of $s \leq d$ distinct primes $p_1, \ldots, p_s$. Then we can also view the set of indices $\mathbf{x}$ as

$$\mathbb{Z}_{q_1} \times \mathbb{Z}_{q_2} \times \cdots \times \mathbb{Z}_{q_d} = G_1 \times G_2 \times \cdots \times G_s,$$

where G_i is the finite Abelian group which is the product of all the $\mathbb{Z}_{q_j}$, $j \in [d]$, such that q_j is a power of p_i.

This canonical tensor product decomposition of $\mathbf{F}$ gives us a natural way to index the rows and columns of $\mathbf{C}$ and each diagonal matrix in $\mathfrak{D}$ using $\mathbf{x}$. More precisely, we index the first half of the rows and columns of $\mathbf{C}$ and every $\mathbf{D}^{[r]}$ in $\mathfrak{D}$ using $(0, \mathbf{x})$; and index the second half of the rows and columns using $(1, \mathbf{x})$, where $\mathbf{x} \in \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_d}$.

With this canonical expression of $\mathbf{F}$ and $\mathbf{C}$, we further inquire the structure of $\mathfrak{D}$, and here one more substantial difficulty awaits us. It turns out that there

are two more properties that we must demand of those diagonal matrices in $\mathfrak{D}$: If $\mathfrak{D}$ does not satisfy these additional properties, then $Z_{\mathbf{C},\mathfrak{D}}(\cdot)$ is #P-hard.

First, for each matrix $\mathbf{D}^{[r]}$ in $\mathfrak{D}$, we use Λ_r and Δ_r to denote the support of $\mathbf{D}^{[r]}$, where Λ_r refers to the first half of the entries and Δ_r refers to the second half of the entries: (We let D_i denote the $(i,i)th$ diagonal entry of $\mathbf{D}$ here)

$$\Lambda_r = \left\{ \mathbf{x} \;\middle|\; D^{[r]}_{(0,\mathbf{x})} \neq 0 \right\} \quad \text{and} \quad \Delta_r = \left\{ \mathbf{x} \;\middle|\; D^{[r]}_{(1,\mathbf{x})} \neq 0 \right\}.$$

We use $\mathcal{S}$ to denote the set of superscripts r such that $\Lambda_r \neq \emptyset$ and $\mathcal{T}$ to denote the set of r such that $\Delta_r \neq \emptyset$, respectively. We can prove that for every $r \in \mathcal{S}$,

$$\Lambda_r = \prod_{i \in [s]} \Lambda_{r,i}$$

must be a direct product of *cosets* $\Lambda_{r,i}$ in the Abelian groups G_i, $i \in [s]$; and for every $r \in \mathcal{T}$,

$$\Delta_r = \prod_{i \in [s]} \Delta_{r,i}$$

must be a direct product of cosets in the same groups; otherwise the problem of computing $Z_{\mathbf{C},\mathfrak{D}}(\cdot)$ is #P-hard.

Second, we prove that for every $r \in \mathcal{S}$ and $r \in \mathcal{T}$, respectively, the diagonal matrix $\mathbf{D}^{[r]}$ on its support Λ_r for the first half of its diagonal entries and on Δ_r for the second half of its diagonal entries, respectively, must possess a *quadratic* structure; Otherwise, $Z_{\mathbf{C},\mathfrak{D}}(\cdot)$ is #P-hard. We can express this quadratic structure as a set of *exponential difference equations* over bases which are appropriate roots of unity of orders equal to various prime powers. The constructions used in this part of the proof are among the most demanding ever attempted.

After all these necessary conditions, we show that if $(\mathbf{C}, \mathfrak{D})$ satisfies all these requirements then there *is* actually a polynomial-time algorithm to compute the function $Z_{\mathbf{C},\mathfrak{D}}(\cdot)$ and thus, the problem of computing $Z_{\mathbf{A}}(\cdot)$ is in P. To this end, we reduce $Z_{\mathbf{C},\mathfrak{D}}(\cdot)$ to $Z_q(\cdot)$ for some appropriate prime power q and as remarked earlier, the tractability of $Z_q(\cdot)$ is new and is of independent interest.

References

1. Dyer, M., Greenhill, C.: The complexity of counting graph homomorphisms. In: Proceedings of the 9th International Conference on Random Structures and Algorithms, pp. 260–289 (2000)
2. Bulatov, A., Grohe, M.: The complexity of partition functions. Theoretical Computer Science 348(2), 148–186 (2005)
3. Goldberg, L., Grohe, M., Jerrum, M., Thurley, M.: A complexity dichotomy for partition functions with mixed signs. In: Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science, pp. 493–504 (2009)
4. Lovász, L.: Operations with structures. Acta Mathematica Hungarica 18, 321–328 (1967)
5. Hell, P., Nešetřil, J.: Graphs and Homomorphisms. Oxford University Press, Oxford (2004)

6. Freedman, M., Lovász, L., Schrijver, A.: Reflection positivity, rank connectivity, and homomorphism of graphs. Journal of the American Mathematical Society 20, 37–51 (2007)
7. Dyer, M., Goldberg, L., Paterson, M.: On counting homomorphisms to directed acyclic graphs. Journal of the ACM 54(6) (2007) Article 27
8. Hell, P., Nešetřil, J.: On the complexity of H-coloring. Journal of Combinatorial Theory, Series B 48(1), 92–110 (1990)
9. Thurley, M.: The complexity of partition functions on Hermitian matrices. arXiv (1004.0992) (2010)
10. Cai, J.Y., Chen, X., Lu, P.: Graph homomorphisms with complex values: A dichotomy theorem. arXiv (0903.4728) (2009)
11. Schaefer, T.: The complexity of satisfiability problems. In: Proceedings of the 10th Annual ACM Symposium on Theory of Computing, pp. 216–226 (1978)
12. Creignou, N., Khanna, S., Sudan, M.: Complexity Classifications of Boolean Constraint Satisfaction Problems. In: SIAM Monographs on Discrete Mathematics and Applications (2001)
13. Feder, T., Vardi, M.: The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. SIAM Journal on Computing 28(1), 57–104 (1999)
14. Valiant, L.: Holographic algorithms. SIAM Journal on Computing 37(5), 1565–1594 (2008)
15. Valiant, L.: Accidental algorthims. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 509–517 (2006)
16. Cai, J.Y., Lu, P.: Holographic algorithms: from art to science. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, pp. 401–410 (2007)
17. Cai, J.Y., Lu, P., Xia, M.: Holant problems and counting CSP. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, pp. 715–724 (2009)
18. Feynman, R., Leighton, R., Sands, M.: The Feynman Lectures on Physics. Addison-Wesley, Reading (1970)
19. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer, New York (1998)
20. Ko, K.: Complexity Theory of Real Functions. Birkhäuser, Boston (1991)
21. Lovász, L.: The rank of connection matrices and the dimension of graph algebras. European Journal of Combinatorics 27(6), 962–970 (2006)
22. Lidl, R., Niederreiter, H.: Finite Fields. . Encyclopedia of Mathematics and its Applications, vol. 20. Cambridge University Press, Cambridge (1997)

Metrical Task Systems and the k-Server Problem on HSTs

Nikhil Bansal[1], Niv Buchbinder[2], and Joseph (Seffi) Naor[3,⋆]

[1] IBM T. J. Watson Research Center, Yorktown Heights, NY 10598
[2] Microsoft Research, Cambridge, MA
[3] Computer Science Dept., Technion, Haifa, Israel

Abstract. We consider the randomized k-server problem, and give improved results for various metric spaces. In particular, we extend a recent result of Coté et al [15] for well-separated *binary* Hierarchically Separated Trees (HSTs) to well-separated d-ary HSTs for poly-logarithmic values of d. One application of this result is an $\exp(O(\sqrt{\log\log k \log n}))$-competitive algorithm for k-server on n uniformly spaced points on a line. This substantially improves upon the prior guarantee of $O(\min(k, n^{2/3})$ for this metric [16].

These results are based on obtaining a refined guarantee for the unfair metrical task systems problem on an HST. Prior to our work, such a guarantee was only known for the case of a uniform metric [5,7,18]. Our results are based on the primal-dual approach for online algorithms. Previous primal-dual approaches in the context of k-server and MTS [2,4,3] worked only for uniform or weighted star metrics, and the main technical contribution here is to extend many of these techniques to work *directly* on HSTs.

1 Introduction

The k-server problem is a central and well studied problem in competitive analysis of online problems and is considered by many to be the "holy grail" problem in the field. The k-server problem is defined as follows. There is a distance function d defined over an n-point metric space and k servers located at the points of the metric space. At each time step, an online algorithm is given a request at one of the points of the metric space, and it is served by moving a server to the requested point. The cost is defined to be the distance traveled by the server. Thus, the goal of an online algorithm is to minimize the total sum of the distances traveled by the servers so as to serve a given sequence of requests. The k-server problem can model many problems, and the most widely studied of these is *paging* with all its variants [27]. Paging is the special case of k-server on a uniform metric. In their seminal paper on competitive analysis, Sleator and Tarjan [27] gave k-competitive algorithms for paging, and also showed that this is the best possible for any deterministic algorithm.

The k-server problem in its full generality was first posed by Manasse et al. [24], who conjectured that a k-competitive online algorithm exists in any metric space and for any

⋆ This work was partly supported by ISF grant 1366/07. Part of this work was done while visiting Microsoft Research, Redmond, WA.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 287–298, 2010.

value of k. This is called the *k-server conjecture*. After a long line of work, a major breakthrough was achieved by Koutsoupias and Papadimitriou [23] who gave a $2k-1$ competitive algorithm. We note that for special metrics such as the uniform metric, line metric [13], and more generally trees [14], a competitive factor of k is known for the k-server problem. The reader is referred to [9] for an excellent survey of various results and history of the problem.

However, despite much interest, randomized algorithms for k-server remain poorly understood. No lower bound better than $\Omega(\log k)$ is known for any metric space, and it is widely believed that an $O(\log k)$-competitive randomized algorithm exists for every metric space against an *oblivious* adversary. This is called the *randomized k-server conjecture*. Yet, better than k-competitive algorithms are known only for few special cases [17, 5, 2, 26, 16, 15]. The most well-studied of these cases is paging and its variants, for which several $\Theta(\log k)$-competitive algorithms are known [17, 1, 2, 21, 20, 8][1]. However, not much is essentially known beyond the uniform metric. Even for the seemingly simple case of n uniformly spaced points on a line, which is perhaps the simplest "non-uniform like" metric, only an $O(n^{2/3})$-competitive algorithm [16] is known, which is $o(k)$-competitive only for $n = o(k^{3/2})$.

Given our current level of understanding, resolving a weaker variant of the randomized k-server conjecture – obtaining a polylog(k, n, Δ)-competitive algorithm for general metrics, where Δ is the diameter of the metric, would be a major breakthrough.[2] Since a general metric space can be embedded into a probability distribution over hierarchically well-separated trees (HSTs) with logarithmic distortion of the distances, it suffices to consider the k-server problem on HSTs to obtain the latter bound. This approach seems particularly promising, as randomized k-server is relatively well understood for uniform metrics, and algorithms on an HST can often be obtained by recursively applying a suitable algorithm on uniform metrics. This has previously been done in many other settings, e.g., metrical task systems and metric labeling [5, 18, 22].

Along these lines, Coté et al. [15] recently gave the first[3] completely formal framework to solving the k-server problem on HSTs by defining a related problem, called the *allocation problem* by [3], on uniform metrics. In the allocation problem (defined formally in Section 2), upon each request, two types of cost are incurred: the *hit-cost* and the *move-cost*. Coté et al. [15] showed that designing an online algorithm that, given any $\epsilon > 0$, has a hit-cost of at most $(1+\epsilon)$ times the total optimum cost and a move-cost of at most poly-logarithmic times the total optimum cost, would imply a poly-logarithmic competitive algorithm for the k-server problem (see Theorem 5 below for a formal statement). Note the asymmetry between the guarantees required for the hit-cost and move-cost. It is crucial that the hit-cost factor is $(1+\epsilon)$, as the hit-cost factor multiplies across the levels of the HST, while the move-cost factor only adds up. In a very interesting result, Coté et al. [15] were able to obtain such a guarantee for the allocation problem on a metric space with *two* points. Using their framework, this guarantee for the allocation problem implies an $O(\log \Delta)$-competitive algorithm for *binary*

[1] Due lack to space here, we refer the reader to [9] for an excellent survey of paging and other results for randomized k-server.

[2] In fact, even an $o(k)$-competitive algorithm would be extremely interesting.

[3] Previously, related (but simpler) problems such as finely competitive paging were studied [8].

HSTs. However, being limited to two points, this result is not general enough to imply any non-trivial guarantees for more general metrics.

Thus, the big outstanding question is to extend the result of Coté et al. [15] to a uniform metric on an arbitrary number of points. However, as already pointed out by [15], it is not even clear how to obtain such a guarantee for a space with three points. In particular, their algorithm crucially uses the fact that the allocation problem on two points looks like a metrical task system problem (defined later) on a line with a special cost structure.

1.1 Our Results

In this work we extend the result of Coté et al. [15] for the allocation problem on any uniform metric on d points. Specifically, we show the following result:

Theorem 1. *There exists an algorithm that is* $(1+\epsilon, O(d \log^2 k \log(k/\epsilon)))$*-competitive for the allocation problem on* d *points.*

Exploring the meaning of such a result in the k-server context, let us define an (ℓ, p, α)-HST as an HST with height ℓ, maximum degree p, and stretch factor α. Given such an HST and combining the above result with Theorem 5 we get the following.

Theorem 2. *Given an* (ℓ, p, α)*-HST metric, there exists an online algorithm for the* k*-server problem with competitive factor:*

$$O\left(\min(\alpha, p\ell \log^2 k \log(k\ell)) \cdot \left(1 + O\left(\frac{p \log^2 k \log(k\ell)}{\alpha}\right)\right)^{\ell+1}\right).$$

In particular, if $\alpha = \Omega\left(\ell p \log^2 k \log(k\ell)\right)$ *(i.e. the HST is well-separated), then the algorithm has a competitive ratio of* $O\left(\ell p \log^2 k \log(k\ell)\right)$.

While our result applies to any number of points d, our competitive ratio with respect to the move-cost linearly varies with d, and hence our result is not strong enough to obtain poly-logarithmic guarantees for general metrics. (For this, one needs a guarantee which is poly-logarithmic in d.) Still, our result has useful consequences. Applying Theorem 2, along with standard embedding results for equally spaced points on the line, we get the following Theorem.

Theorem 3. *There is a* $\exp\left(O(\sqrt{\log \log k \log n})\right)$*-competitive algorithm for* k*-server on* n *equally spaced points on a line. In general, for an arbitrary* n*-point line with diameter* Δ*, our algorithm is* $\exp\left(O(\sqrt{\log \log k \log \Delta})\right)$*-competitive.*

This is the first online algorithm with a sublinear competitive ratio for the line, providing strong evidence that randomization does help for this kind of metric. The line metric is particularly interesting since it is one of the simplest metrics that is essentially different from the uniform metric. Historically, new insights gained from the line case have often been useful for more general metrics. For example, the double coverage algorithm was first discovered for a line [13], and then extended to tree metrics [14]. Obtaining a better online algorithm, even for a line or a circle, is already stated as a major open problem in [9, Problem 11.1] .

Techniques and Comparison with Previous Approaches

Refined Guarantee for Metrical Task Systems: In order to prove Theorem 1, we design here a new algorithm for the *metrical task system* problem (MTS) with refined guarantees, which are interesting on their own. MTS was introduced by Borodin et al. [10] as a generalization of various online problems. In this problem there is a machine, or server, which can be in any one of n states $1, 2, \ldots, n$, and a metric d defining the cost of moving between states. At each time step t, a new task appears and needs to be served. The new task is associated with a cost vector $c_t = (c_t(1), \ldots, c_t(n))$ denoting the processing cost in each of the states. To serve the task, the machine is allowed to move to any other state from its current state. Assuming the machine is currently in state i, and it moves to state j, the cost of serving the task is its processing cost $c_t(j)$ plus the movement cost d_{ij}. The goal is to minimize the total cost.

We obtain the following refined guarantee for MTS on an HST.

Theorem 4. *Consider the MTS problem on an HST of height ℓ and n points. For any $0 < \gamma < 1$, there is an online algorithm achieving the following bounds:*

- *Its service cost is bounded by $(1 + \gamma)$ times the optimal cost.*
- *Its movement cost is bounded by $O(\log(\frac{n}{\gamma}) \cdot \min\{\ell, \log(\frac{n}{\gamma})\})$ times the optimal cost.*

While poly-logarithmic guarantees for randomized MTS on HSTs (and hence general metrics) are already known [5,7,18], the key difference here is that the competitive ratio with respect to the service cost is only $(1 + \gamma)$. Previously, such a result was only known for uniform metrics [5,7,19], where this guarantee was referred to as the *unfair* MTS guarantee. Thus, Theorem 4 can be viewed as an extension of the unfair MTS result of [5,7,19] to HSTs. Such an extension is crucial to obtain meaningful results for the allocation problem (see Section 4.1 for more details about this reduction). In fact, obtaining such a result was already proposed as an approach to attacking the allocation problem [25] (see e.g. slide 45).

As we explain below, it is quite unclear whether Theorem 4 can be obtained using previous techniques. Next, we explain the techniques we use here.

Previous Results on Randomized MTS: In a breakthrough paper, Bartal et al. [5] obtained the first polylogarithmic competitive randomized algorithm for HSTs (and hence general metrics) by recursively solving a stronger variant of the MTS problem on uniform metrics. This stronger variant is the unfair MTS problem that, given any $\gamma > 0$, pays a movement cost of $O(\log(\frac{n}{\gamma}))$ times the optimum, but has service cost only $(1+\gamma)$ times the optimum. In particular, previous algorithms obtaining a polylogarithmic competitive ratio on HSTs, achieved it by recursively "combining unfair MTS algorithms" on uniform metrics.

A natural question is whether the approach of recursively combining an unfair MTS algorithm for uniform metrics can be used to obtain an unfair MTS algorithm for the entire HST. However, it is not clear how to use previous techniques toward this task [6]. In particular, any approach based on "combining unfair MTS" loses right away at the second level of the HST the distinction between the service costs and movements costs. This happens since the cost vector for the unfair MTS algorithm running at a node at a

higher level must necessarily use the total cost incurred at the lower level (i.e. the sum of service costs plus movement costs incurred at the lower level). Thus, it is not clear how to even track the service costs and movement costs for HSTs with 2 or more levels.

Our Techniques: We completely move away from recursively combining unfair MTS algorithms on uniform metrics. In particular, our algorithm is a "one shot algorithm" that works directly on the whole metric, and is based on linear programming techniques. Specifically, we cast the (offline) MTS problem on an HST as a minimization linear program. We then obtain the dual maximization problem which is used to bound the cost of our online solution. When a new cost vector arrives, a set of equations is solved to compute the amount of "flow" it should move from each leaf in the HST to other leaves. This flow solution is dictated by a single decision implied by a "connecting function" between primal and dual variables.

At a high level, our approach follows the general primal-dual framework for designing online algorithms developed by Buchbinder and Naor [11,12,2,4,3]. However, the main contribution here is that we extend many of the previous techniques to work for HSTs. In particular, all previous results and techniques [2,4,3] hold only for uniform metrics or weighted star metrics[4]. These metrics seem substantially simpler than HSTs, and in general it is not at all clear how to extend these results from uniform metrics to HSTs. For example, several ways are known to obtain poly-logarithmic guarantees for k-server on the uniform metric (paging), but extending these results to HSTs would prove the weaker randomized k-server conjecture!

2 Preliminaries

Allocation Problem: The allocation problem is defined as follows. There is a uniform metric on n points and there are up to k available servers. At time step t, the total number of available servers $k(t) \leq k$ is specified, and a request arrives at some point i_t. The request is specified by a $(k+1)$-dimensional vector $\boldsymbol{h}^t = (h^t(0), h^t(1), \ldots, h^t(k))$, where $h^t(j)$ denotes the cost when serving the request using j servers. Upon receiving a request, the algorithm may choose to move additional servers to the requested point and then serve it. The cost is divided into two parts. The *Move-Cost* incurred for moving the servers, and the *Hit-Cost*, $h^t(j)$, determined by the cost vector. The goal is to minimize the total cost. In addition, the cost vectors at any time are guaranteed to satisfy the following *monotonicity* property: for any $0 \leq j \leq k-1$, the costs satisfy $h^t(j) \geq h^t(j+1)$. That is, serving a request with less resources costs more.

Denote by $\mathrm{Optcost}$ the optimal cost of an instance of the allocation problem. Coté et al. [15] showed that if there is an online algorithm that incurs a hit cost of $(1+\varepsilon)\mathrm{Optcost}$ and a move cost of $\beta(\epsilon)\mathrm{Optcost}$, where $\beta(\cdot)$ is a polylogarithmic function, then there is a polylogarithmic competitive algorithm for the k-server problem on general metrics. More generally, the next theorem implicitly follows from their work.

[4] In related work, [3] shows how an unfair MTS algorithm on uniform metrics (i.e. the result of [5]) can be obtained using the primal-dual framework. It also solves the finely competitive paging problem [8], and the allocation problem assuming the costs satisfy a certain convexity assumption.

Theorem 5 ([15,3]). *Suppose there is a $(1+\epsilon, \beta(\epsilon))$-competitive algorithm for the allocation problem on a uniform metric on d points. Let H be a d-ary HST with depth ℓ and parameter α. Then, for any $\epsilon \leq 1$, there is a $\beta(\epsilon)\gamma^{\ell+1}/(\gamma-1)$-competitive algorithm for the k-server problem on H, where*

$$\gamma = (1+\varepsilon)\left(1+\frac{3}{\alpha}\right) + O\left(\frac{\beta(\epsilon)}{\alpha}\right).$$

Hierarchically well-separated trees: We formally define a hierarchically well-separated tree (HST) with stretch factor α and fix some notation. We denote the leaves of an HST by $1,\ldots,n$ and use i to index them. In an HST all leaves have the same depth, denoted by ℓ (where we use the convention that a star has depth 1). The root r is at level 0. Let $\ell(v)$ denote the level of node v. Then, the distance of v to its parent is $\alpha^{\ell-\ell(v)}$ and so the diameter of the HST is $O(\alpha^{\ell-1})$. Let T_v denote the set of leaves in the subtree rooted at node v. For a leaf i, let (i,j) denote the j-th ancestor of i. That is, $(i,0)$ is i itself, $(i,1)$ is the parent of i, and so on. Note that (i,ℓ) is the root r for any leaf i. Let $p(v)$ be the parent node of v and let $C(v)$ be the set of children of v. The number of children of v is denoted by $|C(v)|$; if v is a leaf then we use the convention $|C(v)|=1$.

LP Formulation for MTS on an HST: We study the MTS problem on a metric $\mathcal{M}$ defined by an HST. By a standard transformation, we can assume that the cost vector at any time has the form $c \cdot e_i$, where c is arbitrarily small and $e_i = (0,0,\ldots,1,0,\ldots)$ has 1 in the i-th position. Hence, we use i_t to denote the location with non-zero cost at time t.

We now present our linear programming formulation of the MTS problem on an HST. Our algorithm keeps a fractional solution in which each leaf i has mass $y_{i,t}$ at time t. Let $y_{v,t}$ denote the total mass in the subtree rooted at v at time t. Then, by definition, $y_{v,t} = \sum_{w\in C(v)} y_{w,t} = \sum_{i\in T_v} y_{i,t}$. Note that this is consistent with the definition for leaves. The variables will be $y_{i,t}$ for each leaf i and time t. Let $z_{v,t}$ denote the decrease in the total probability mass in leaves of T_v at time t. Note that there is no need to define a variable $z_{r,t}$ (for the root) and without loss of generality it suffices to charge only for removing mass from a subtree (and not for introducing more mass to a subtree). The linear program is as follows.

$$(P) \qquad \min \quad \sum_{v,t} \alpha^{\ell-\ell(v)} z_{v,t} + \sum_t c \cdot y_{i_t,t}$$

$$\text{For any time } t: \qquad \sum_i y_{i,t} = 1 \tag{1}$$

$$\text{For any time } t \text{ and subtree } T_v\ (v \neq r): \qquad z_{v,t} \geq \sum_{i\in T_v}(y_{i,t-1} - y_{i,t}) \tag{2}$$

Clearly, the formulation is valid. The first constraint says that the total probability mass on the leaves is exactly 1. The second set of constraints captures the movement cost (i.e. the cost of decreasing the total probability mass in a subtree) at each internal node.

We now define the dual program. We associate variables a_t and $b_{v,t}$ with the above constraints. For notational convenience, let $\Delta b_{v,t+1} = b_{v,t+1} - b_{v,t}$. If v is the j-th

parent of leaf i, we will use the notation v and (i,j) interchangeably (especially if we need to specify the relation of v to i).

$$(D) \qquad \max \quad \sum_t a_t$$

$$\text{For any time } t \text{ and leaf } i \neq i_t: \qquad a_t - \sum_{j=0}^{\ell-1} \Delta b_{(i,j),t+1} \leq 0 \tag{3}$$

$$\text{For any time } t \text{ and leaf } i_t: \qquad a_t - \sum_{j=0}^{\ell-1} \Delta b_{(i_t,j),t+1} \leq c \tag{4}$$

$$\text{For any time } t \text{ and subtree } T_v: \qquad b_{v,t} \leq \alpha^{\ell-\ell(v)} \tag{5}$$

Equivalently, the last constraint can be written as $b_{(i,j),t} \leq \alpha^j$ for any time t, leaf i, and index j.

We view the dual as follows. With each node v in the tree there is a variable b_v that varies as a function of time, but is bounded by a fixed constant depending on the level of v (constraint (5)). The dual profit can only be obtained by increasing a_t, and hence we try to increase these variables as much as possible over time. At time t, when a request arrives at leaf i_t, constraints (3) require that for every leaf $i \neq i_t$, the sum of b_v values along the path from i to the root must increase so as to offset a_t. Thus, they force the b_v variables to increase as a_t is raised. However, as the b_v's are bounded from above, this process cannot go on for too long, thus preventing the growth of the dual profit. The rescue comes from the slack in the corresponding constraint for leaf i_t (constraint (4)), allowing us to decrease the appropriate b_v variables as requests arrive over time.

3 The Metrical Task System Algorithm

The algorithm for the MTS problem on an HST is based on a two-step approach. First, we show how to maintain online a fractional solution, then show how to transform the fractional online algorithm into a randomized algorithm (the second step is actually trivial). The randomized algorithm is going to maintain a primal solution (a probability distribution on states) and a corresponding dual solution. The high level idea of the algorithm is very simple. It maintains a (carefully chosen) relation between the primal and dual variables. When a cost arrives at leaf i_t at time t, we update the dual variables subject to the dual constraints ((3)-(5)). The relation between the primal and the dual variables determines how much probability mass should be moved out of leaf i_t and how it should be distributed among other leaves.

The relation between the dual and primal variables is that the value of variable $y_{v,t}$ (amount of mass in subtree T_v) is a function of the dual variable $b_{v,t+1}$:

$$y_{v,t} \triangleq f(b_{v,t+1}) \triangleq \frac{\gamma \cdot |C(v)|}{n} \left(\exp\left(\frac{\ln\left(1 + n/\gamma\right) b_{v,t+1}}{\alpha^{\ell-\ell(v)}} \right) - 1 \right). \tag{6}$$

This relation is maintained throughout the execution of the algorithm. Let $0 < \gamma < 1$ be an arbitrary constant. Recall that if v is a leaf then $|C(v)| = 1$. Before continuing with the description of the algorithm, we should derive several properties from the invariant. These properties are somewhat technical and we omit them due to space limitations.

3.1 The Algorithm

We are now ready to describe our algorithm. At any time t, the algorithm maintains a distribution on the leaves $y_{i,t}$. We describe how this distribution is updated upon arrival of a new request. Suppose, without loss of generality, that the request at time t arrives at leaf 1 with cost $(c, 0, 0, ..., 0)$. The high level idea of the algorithm is very simple. Since the cost is non-zero only at leaf 1, we should move some probability mass out of leaf 1, and distribute it to various other leaves. To determine the exact amount at each leaf, the algorithm simply should keep the following invariants:

MTS Conceptual Algorithm: When a request arrives at time t at leaf 1 keep the following invariants:

1. **(Tight Duals:)** All dual constraints of type (3) on leaves $2, 3, \ldots, n$ are tight.
2. **(Hit leaf:)** Either $y_{1,t} = 0$, or the dual constraint (4) on leaf 1 is tight.
3. **(Consistency:)** For each node v, $y_v = \sum_{w \in C(v)} y_w$.

It turns out that the conceptual algorithm above completely determines how much mass we should move from leaf 1 to any other leaf in the sub-tree. So, essentially, the restriction on keeping the dual constraints tight and keeping consistency (i.e., mass in each sub-tree is equal to the mass in its children) completely determines the algorithm!

For the rest of this section we perform some calculations to give an explicit description of the algorithm, specifying the exact rate at which we should move mass in the tree. Having obtained these rates, we will bound the total movement cost and the total service cost by relating it to the growth of the dual profit.

We first give an intuitive description of the process. When the cost arrives at leaf 1 at time t, if $y_{1,t} = 0$ then the primal cost is zero and the dual profit is also zero. Moreover, nothing changes and all the invariants continue to hold. Thus, we consider the case that $y_{1,t}$. We start increasing a_t at rate 1. At each step we would like to keep all dual constraints tight, as well as consistent. However, raising a_t violates the dual constraints on leaves $2, 3, \ldots, n$, forcing us to raise other dual variables to keep these constraints tight. Raising these variables may also violate consistency (Invariant (3)), and thus lead to the update of other dual variables. This process results in the transfer of mass from leaf 1 to leaves $2, 3, \ldots, n$, and also the constraint on leaf 1 becomes tighter. We stop the updating process when either the constraint on leaf 1 is tight, or when $y_{1,t} = 0$. We next consider this process more formally.

Consider the root, and let 1_j, $1 \leq j \leq \ell$, denote the node at level j containing leaf 1 (1_ℓ is the leaf itself). Since mass moves out of each subtree containing 1, variable $b_{v,t+1}$ decreases for each $v = 1_j$. For every other node v in the tree not containing 1, the probability mass increases. Any node v not containing 1 must be a sibling of some unique node 1_j. By symmetry, all siblings v of a node 1_j must increase $b_{v,t+1}$ at the same rate (it can be easily checked that this is indeed necessary to keep consistency). We wish to determine the increase/decrease rate of all dual variables in the tree with respect to a_t. Let us use the following notation. For $1 \leq j \leq \ell$, let $\Delta b_j \triangleq -\frac{db_{1_j,t+1}}{da_t}$ be the rate at

which $b_{1_j,t+1}$ is *decreasing* with respect to a_t. For $1 \leq j \leq \ell$, let $\Delta b'_j \triangleq \frac{db_{w,t+1}}{da_t}$ be the rate at which the siblings of 1_j are *increasing* with respect to a_t.

We can derive the quantities Δb_j and $\Delta b'_j$ in a top-down fashion as follows (the exact arguments are omitted due to lack of space). We first consider the siblings of 1_1 (i.e. children of root other than 1_1). Let v be one of these siblings. If we raise $b_{v,t+1}$ by $\Delta b'_1$, by the consistency requirement, the sum of $\Delta b'$ on any path from v to a leaf in T_v must be $\delta(1) \cdot \Delta b'_1$. As a_t is growing at rate 1, keeping the dual constraint (3) tight for leaves in T_v requires that

$$\delta(1) \cdot \Delta b'_1 = 1.$$

This takes care of the dual constraints for these leaves. Now, this increase of mass must come by decreasing the mass in T_{1_1} since the total probability mass is 1. To keep consistency of the root we should set Δb_1 so that:

$$\Delta b'_1 = (\Delta b_1 + \Delta b'_1) \cdot \left(y_{1_1,t} + \frac{\gamma \cdot |C(1)|}{n} \right) / (1+\gamma).$$

We repeat the argument for siblings of node 1_2. Let v be a sibling. Consider a path from a leaf in T_v to the root. Their dual constraint (3) already grows at rate $1 + \delta(1)\Delta b_1$. This must be compensated by increasing $b_{v,t+1}$, and by consistency all the variables $b_{w,t+1}$ for $w \in T_v$. Therefore, $\Delta b'_2$ has to be set so that:

$$\delta(2) \cdot \Delta b'_2 = 1 + \delta(1)\Delta b_1.$$

Again, this additional increase of mass must come from an additional decreasing mass in T_{1_2}. To keep consistency of 1_1 we must set Δb_2 so that:

$$\Delta b'_2 = (\Delta b_2 + \Delta b'_2) \cdot \left(y_{1_2,t} + \frac{\gamma \cdot |C(2)|}{n} \right) / \left(y_{1_1,t} + \frac{\gamma \cdot |C(1)|}{n} \right).$$

Continuing on, we obtain a system of linear equations. Due to lack of space, the set of constraints fully defining our algorithm is omitted. Solving the equations we get:

$$\Delta b'_j = \frac{(1+\gamma)}{\delta(j)} \cdot \left(y_{1_{j-1},t} + \frac{\gamma \cdot |C(j-1)|}{n} \right)^{-1}$$

$$\Delta b_j = \frac{1+\gamma}{\delta(j)} \cdot \left(\left(y_{1_j,t} + \frac{\gamma \cdot |C(j)|}{n} \right)^{-1} - \left(y_{1_{j-1},t} + \frac{\gamma \cdot |C(j-1)|}{n} \right)^{-1} \right).$$

This fully defines our explicit MTS algorithm in terms of derivatives. To implement it, the algorithm simply does a binary search for the correct value of a_t defining the correct flow.

MTS Explicit Algorithm: When a request arrives at time t at leaf 1 keep the following invariants:

1. While the dual constraint of leaf 1 is not tight and $y_{1,t} > 0$:
2. Increase a_t with rate 1.
3. Decrease each b_{1_j} with rate:

$$\frac{db_{1_j,t+1}}{da_t} = \frac{1+\gamma}{\delta(j)} \cdot \left[\left(y_{1_j,t} + \frac{\gamma \cdot |C(j)|}{n} \right)^{-1} - \left(y_{1_{j-1},t} + \frac{\gamma \cdot |C(j-1)|}{n} \right)^{-1} \right].$$

4. For each sibling w of 1_j increase $b_{w,t+1}$ with rate:

$$\frac{db_{w,t+1}}{da_t} = \frac{(1+\gamma)}{\delta(j)} \left(y_{1_{j-1}} + \frac{\gamma \cdot |C(j-1)|}{n} \right)^{-1}.$$

5. For every node w, recursively, top to bottom, if the variable of the parent of w, $b_{v,t+1}$ is increasing/decreasing with rate c, then decrease/increase $b_{w,t+1}$ with rate c/α.

Based on these rates, we can now calculate the movement cost and the service cost as a function of the dual profit. This allows us to show our main result, Theorem 4. The proof is omitted.

4 Applications of the MTS Algorithm

In this section we will describe applications of our method to several problems.

4.1 The Allocation Problem

The allocation problem on d points and k servers can be viewed as an MTS problem on $O((k+1)^d)$ states. There is a state for each possible way of distributing (up to) k servers among the d points. There is a natural metric on these states (equal to half the hamming distance between vectors corresponding to two states). The diameter (ratio of maximum to minimum distance between any two distinct points) of this space is k.

One issue in the allocation problem is that the number of available servers $k(t)$ can change with time. This can be handled as follows. Suppose we have an instance of the allocation problem on d points. Now, imagine that there are $d+1$ points and the number of servers is fixed at k. The number of servers at point $d+1$ is supposed to be at least $k - k(t)$. We can enforce this via MTS, by setting an infinite cost to states having less than $k - k(t)$ servers at $d+1$. Thus, the number of states in MTS is at most $O((k+1)^{d+1})$.

We embed the metric space of the MTS problem into a probability distribution over dominating HSTs. Note that the embedding only affects the move costs. Moreover, the distortion is at most $O(\log \Delta) = O(\log k)$, where Δ is the diameter of the space. The depth of the HST is $\ell = O(\log k)$. Theorem 4 implies an $(1+\epsilon, O(\ell \log(n/\epsilon)))$

competitive algorithm for MTS on an HST with depth ℓ. By the application to an allocation problem on d points, in which $n = O(k^d)$ and adding an addition factor $\log(\Delta) = O(\log k)$ due to distortion in the movement costs, we obtain that

Theorem 6. *There exists an* $(1+\epsilon, O(d \log^2 k \log(k/\epsilon)))$*-competitive algorithm for the allocation problem of degree* d*.*

4.2 Application to k-Server on d-Ary HSTs

We use theorem 5 that relates the k-server problem on HSTs to the allocation problem. By Theorem 6 we have $\beta(\epsilon) = O(d \log^2 k \log(k/\epsilon)))$. Thus we obtain:

Theorem 7. *Let* T *be a* d*-ary HST with depth* ℓ *and parameter* α*, then for any* $\epsilon \leq 1$*, there exists a competitive algorithm for the HST with competitive factor:*

$$O\left(\min(\alpha, d\log^2 k \log(k/\epsilon)/\epsilon) \cdot \left(1+\epsilon+O(\frac{d\log^2 k\log(k/\epsilon)}{\alpha})\right)^{\ell+1}\right).$$

Choosing $\epsilon = \frac{1}{\ell}$*, if* $\alpha = \Omega\left(\ell d \log^2 k \log(k\ell)\right)$*, the algorithm is* $O\left(\ell d \log^2 k \log(k\ell)\right)$*-competitive.*

4.3 The k-Server Problem on Equally Spaced Points on the Line

A well known (folklore) result for embedding a line into an HST is the following[5].

Lemma 1. *For any* $\alpha \geq 2$*, the line metric can be embedded into an* $(\ell = \log \Delta / \log \alpha, d = O(\alpha), \alpha)$*-HST, with distortion of* $\alpha \log \Delta$*.*

Apply Theorem 7 with $\epsilon = 1$ and $\alpha = \exp(O(\sqrt{\log\log k \log \Delta}))$ (which turn out to optimal choices of parameters), yielding:

Theorem 8. *There exists an* $\exp\left(\sqrt{O(\log\log k \log \Delta)}\right)$*-competitive algorithm for the the* k*-server problem on the line.*

References

1. Achlioptas, D., Chrobak, M., Noga, J.: Competitive analysis of randomized paging algorithms. Theory Computer Science 234(1-2), 203–218 (2000)
2. Bansal, N., Buchbinder, N., Naor, J.S.: A primal-dual randomized algorithm for weighted paging. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, pp. 507–517 (2007)
3. Bansal, N., Buchbinder, N., Naor, S.: Towards the randomized k-server conjecture: A primal-dual approach. In: ACM-SIAM Symposium on Discrete Algorithms, SODA 2010 (to appear, 2010)

[5] For example, this can be done by recursively splitting a line of diameter Δ in random into d pieces of expected diameter $O(\Delta/d)$.

4. Bansal, N., Buchbinder, N., Naor, J.(S).: Randomized competitive algorithms for generalized caching. In: Proceedings of the 40th annual ACM symposium on Theory of computing, pp. 235–244 (2008)
5. Bartal, Y., Blum, A., Burch, C., Tomkins, A.: A polylog(n)-competitive algorithm for metrical task systems. In: Proceedings of the 29th Annual ACM Symposium on Theory of computing, pp. 711–719 (1997)
6. Blum, A.: Personal communication
7. Blum, A., BurchP, C.: On-line learning and the metrical task system problem. Machine Learning 39(1), 35–58 (2000)
8. Blum, A., Burch, C., Kalai, A.: Finely-competitive paging. In: IEEE Symposium on Foundations of Computer Science, pp. 450–458 (1999)
9. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press, Cambridge (1998)
10. Borodin, A., Linial, N., Saks, M.E.: An optimal on-line algorithm for metrical task system. Journal of the ACM 39(4), 745–763 (1992)
11. Buchbinder, N., Naor, J.: Online primal-dual algorithms for covering and packing problems. In: Proceedings of the 13th Annual European Symposium on Algorithms (2005)
12. Buchbinder, N., Naor, J.: The design of competitive online algorithms via a primal-dual approach. Foundations and Trends in Theoretical Computer Science 3(2-3), 93–263 (2009)
13. Chrobak, M., Karloff, H., Payne, T., Vishwanathan, S.: New results on server problems. SIAM Journal on Discrete Math 4(2), 172–181 (1991)
14. Chrobak, M., Larmore, L.: An optimal on-line algorithm for k-servers on trees. SIAM Journal on Computing 20(1), 144–148 (1991)
15. Coté, A., Meyerson, A., Poplawski, L.: Randomized k-server on hierarchical binary trees. In: STOC 2008: Proceedings of the 40th annual ACM symposium on Theory of computing, pp. 227–234 (2008)
16. Csaba, B., Lodha, S.: A randomized on-line algorithm for the k-server problem on a line. Random Structures and Algorithms 29(1), 82–104 (2006)
17. Fiat, A., Karp, R.M., Luby, M., McGeoch, L.A., Sleator, D.D., Young, N.E.: Competitive paging algorithms. Journal of Algorithms 12(4), 685–699 (1991)
18. Fiat, A., Mendel, M.: Better algorithms for unfair metrical task systems and applications. SIAM Journal on Computing 32(6), 1403–1422 (2003)
19. Fiat, A., Mendel, M.: Better algorithms for unfair metrical task systems and applications. SIAM Journal on Computing 32(6), 1403–1422 (2003)
20. Irani, S.: Randomized weighted caching with two page weights. Algorithmica 32(4), 624–640 (2002)
21. Irani, S.: Page replacement with multi-size pages and applications to web caching. In: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, pp. 701–710 (1997)
22. Kleinberg, J.M., Tardos, É.: Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. J. ACM 49(5), 616–639 (2002)
23. Koutsoupias, E., Papadimitriou, C.H.: On the k-server conjecture. Jornal of the ACM 42(5), 971–983 (1995)
24. Manasse, M.S., McGeoch, L.A., Sleator, D.D.: Competitive algorithms for server problems. Journal of Algorithms 11(2), 208–230 (1990)
25. Meyerson, A.: http://www.cs.ucla.edu/~awm/talks/kserver.ppt
26. Seiden, S.S.: A general decomposition theorem for the k-server problem. In: Proceedings of the 9th Annual European Symposium on Algorithms, pp. 86–97 (2001)
27. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of the ACM 28(2), 202–208 (1985)

Scheduling Periodic Tasks in a Hard Real-Time Environment*

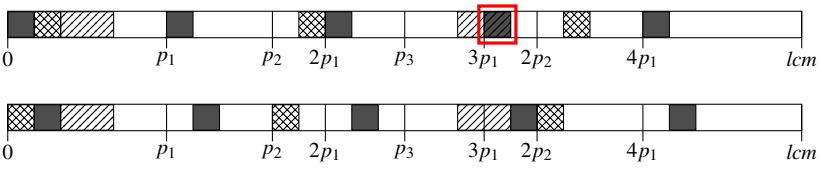
Friedrich Eisenbrand[1], Nicolai Hähnle[1], Martin Niemeier[1], Martin Skutella[2], José Verschae[2], and Andreas Wiese[2]

[1] EPFL, Lausanne, Switzerland
[2] TU Berlin, Germany

Abstract. We give a rigorous account on the complexity landscape of an important real-time scheduling problem that occurs in the design of software-based aircraft control. The goal is to distribute tasks $\tau_i = (c_i, p_i)$ on a minimum number of identical machines and to compute offsets a_i for the tasks such that no collision occurs. A task τ_i releases a job of running time c_i at each time $a_i + k \cdot p_i$, $k \in \mathbb{N}_0$ and a collision occurs if two jobs are simultaneously active on the same machine. Our main results are as follows: (i) We show that the minimization problem cannot be approximated within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$. (ii) If the periods are harmonic (for each i, j one has $p_i \mid p_j$ or $p_j \mid p_i$), then there exists a 2-approximation for the minimization problem and this result is tight, even asymptotically. (iii) We provide asymptotic approximation schemes in the harmonic case if the number of different periods is constant.

1 Introduction

The motivation for this research comes from a real-word combinatorial optimization problem that was communicated to us by our industrial partner, a major avionics company. The aircraft designers need to schedule highly critical periodic control tasks with a predictable and static scheduling policy such that preemption and dynamic effects are avoided. The model that is used in this context is as follows. One is given *tasks* $\tau_1, \ldots, \tau_n$ where each task $\tau_i = (c_i, p_i)$ is characterized by its *execution time* $c_i \in \mathbb{N}$ and *period* $p_i \in \mathbb{N}$. The goal is to assign the tasks to identical machines and to compute offsets $a_i \in \mathbb{N}_0$ such that no collision occurs. A task τ_i generates one *job* with execution time c_i at every time unit $a_i + p_i \cdot k$ for all $k \in \mathbb{N}_0$. Each job needs to be processed immediately and non-preemptively after its generation on the task's machine. A collision occurs if two jobs are simultaneously active on one machine.



* This work was partially supported by Berlin Mathematical School, by DFG research center MATHEON, by the DFG Focus Program 1307 within the project "Algorithm Engineering for Real-time Scheduling and Routing", and by the Swiss National Science Foundation.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 299–311, 2010.

The picture above shows three tasks $\tau_1 = (1,6), \tau_2 = (1,10)$ and $\tau_3 = (2,15)$. The upper part shows an infeasible assignment of offsets ($a_1 = 0,\ a_2 = 1,\ a_3 = 2$) whereas the lower part shows a feasible assignment of offsets ($a_1 = 1,\ a_2 = 0,\ a_3 = 2$). Notice that the schedule repeats after the least-common multiple (*lcm*) of the periods.

In the single machine context and with unit execution times, this problem was studied by Wei and Liu [WL83] who called the problem of computing offsets the *periodic maintenance problem*. Baruah et al. [BRTV90] and independently [BBNS02, Bha98] show that the periodic maintenance problem is NP-hard in the strong sense.

The engineers are however interested in the corresponding *machine minimization* problem, i.e., they want to find the minimum number of identical machines on which the tasks can be distributed in a feasible way. We refer to this problem as the *periodic maintenance minimization problem*. Korst et al. [KALW91, KAL96] studied this problem and show independently from [BRTV90] and [BBNS02, Bha98] that it is NP-hard in the strong sense. It occurs in particular when additional features have to be implemented on a given architecture. For the avionics company it is then preferable to re-design the control software only rather than to re-design and change the hardware components. Sometimes even moderately sized real-word instances turn out to be unsolvable with state-of-the-art integer programming approaches. One feature that the easier instances share is that, with only few exceptions, their tasks have harmonic periods, i.e., for each pair of tasks τ_i, τ_j one has $p_i \mid p_j$ or $p_j \mid p_i$. Thus, one question that arises is whether instances with this divisibility property are easier to solve than general instances.

For many combinatorial optimization problems, the answer to an analogous question is indeed *yes*. For instance, KNAPSACK with divisible items [Mar85, VA97] or the MIXING SET problem with divisible capacities [ZIRdF08, CSW08] are only two examples, where one has polynomial time algorithms on the one hand for the divisible case and NP-hardness for the general case. For the periodic maintenance minimization problem that is in the focus of this work the difference is however drastic and reflects very well the experience with those real-word instances whose tasks mostly have harmonic periods and very general instances.

Our contribution is a rigorous account on the complexity and approximability landscape of the above described machine-minimization problem. Our results are summarized in Table 1. In this extended abstract, we limit ourselves to the description of the following results that are highlighted in this table. More results and details are presented in the full version of this paper [EHN+10].

► We prove that, for any $\varepsilon > 0$, it is NP-hard to approximate the periodic maintenance minimization problem within a factor of $n^{1-\varepsilon}$, i.e., that the trivial approximation algorithm is essentially tight. This explains the difficulty of moderately sized instances without harmonic periods from a theoretical viewpoint. The result is achieved by a reduction from COLORING that relies on basic number-theoretic results like the Chinese Remainder Theorem and the Prime Number Theorem. We remark that the reduction has been given independently in [BBNS02, Bha98]. The hardness result also holds under resource augmentation.

► We show that the periodic maintenance minimization problem with harmonic periods allows for a 2-approximation algorithm. Furthermore, we show that this is tight, even asymptotically. It is remarkable that a simple variant of First-Fit can be analyzed to

Table 1. The approximability landscape of the periodic maintenance minimization problem. Here q_k and q_1 denote the largest and smallest period length, respectively.

arbitrary periods		
period lengths k	algorithms	hardness results
k arbitrary	$2OPT + k - 1$	$n^{1-\varepsilon} OPT$
k constant	$\left(\frac{3}{2} + \varepsilon\right) OPT + k$	$\left(\frac{3}{2} - \varepsilon\right) OPT + k - 1$

harmonic periods		
period lengths k	algorithms	hardness results
k arbitrary	$2OPT$	$(2-\varepsilon)\, OPT + o(OPT)$
k constant	$(1+\varepsilon)\, OPT + k$	$\left(\frac{3}{2} - \varepsilon\right) OPT + k - 1$
q_k/q_1 constant	$(1+\varepsilon)\, OPT + 1$	$(2-\varepsilon)\, OPT$

be a 2-approximation algorithm. The analysis differs however considerably from the simple analysis that shows that First-Fit for BIN-PACKING yields a 2-approximation. The novel concept that we use is the one of a witness for certain groups of machines. These witnesses prove that these groups are heavily loaded. If a witness for a certain group of machines is missing, the instance can be separated into two independent sub-instances. This allows for an analysis by induction.

▶ Even though the 2-approximation result for the case of harmonic periods is tight, we show that a stronger restriction leads to an asymptotic PTAS: If the number of different periods k is constant, we have an efficient algorithm with approximation guarantee $(1+\varepsilon)OPT + k$, for any constant $\varepsilon > 0$. The basic approach follows the ideas of the classical APTAS for BIN-PACKING of Fernandez de la Vega and Luecker [FdlVL81]. The more complicated nature of the periodic maintenance problem, however, requires several interesting and nontrivial extensions of the techniques such as a more sophisticated rounding procedure and advanced structural insights into the solutions to the periodic maintenance problem. This helps to enumerate the solution space to find a template that can be turned into a solution with the desired approximation guarantee.

Related work. There is excessive literature on real-time scheduling; for surveys see, e.g., [BHR93, But04, Leu04]. In contrast to our problem, one is often interested in verifying whether a set of tasks can be scheduled on one machine with a certain scheduling policy. The periodic maintenance minimization problem generalizes BIN-PACKING. In fact, if the periods of tasks are all identical, the problem is equivalent to BIN-PACKING. The problem is, however, more general and the approximation ratios known for BIN-PACKING do not carry over. While there is, for instance, a 1.5-approximation algorithm for BIN-PACKING [SL94], achieving this performance ratio is impossible for our problem, even for the case of harmonic periods.

2 Harmonic Periods

First-Fit is a simple and very popular heuristic for many packing problems, such as, e.g., BIN-PACKING etc. Adapted to our problem, First-Fit considers tasks in some given

order and greedily packs the current task on the first open machine on which it fits. In case there is no such machine, it opens a new machine on which the current task is scheduled.

A crucial subproblem occurring in this context is to decide whether a task τ can be scheduled on a machine on which other tasks, say $\tau_1, \ldots, \tau_n$, have already been scheduled without changing the offsets of these tasks. If all processing times are equal to 1 and the period of τ is p, we will see in Section 3 that the feasible offsets a for task τ are the solutions to the system

$$a \not\equiv a_j \bmod \gcd(p, p_j) \quad \text{for } j = 1, \ldots, n. \tag{1}$$

For arbitrary (not harmonic) $p_j \in \mathbb{N}$ and $p = \prod_{i=1}^{n} p_j$, this is the NP-complete problem of computing *simultaneous incongruences*, see, e.g., [GJ79]. Thus, for instances of the periodic maintenance minimization problem with arbitrary periods, already this crucial subproblem is NP-hard and it is not clear how to implement First-Fit.

In this section we consider the special case of harmonic periods. In Section 2.1 we show how the crucial subproblem can be solved efficiently in this case by making clever use of a special solution structure.

In the following we always assume that tasks $\tau_1 = (c_1, p_1), \ldots, \tau_n = (c_n, p_n)$ are sorted such that $p_j \mid p_{j+1}$, for $j = 1, \ldots, n-1$. Moreover, the number of different period lengths of the input-tasks is denoted by k and the set of all task periods is denoted by $Q = \{q_1, \ldots, q_k\}$, where $q_i \mid q_{i+1}$, for $i = 1, \ldots, k-1$.

2.1 Bin Trees

An important encoding of single-machine schedules which we use repeatedly in this paper is the (compressed) *bin tree* that we now explain. Assume that offsets a_j for tasks $\tau_j = (c_j, p_j)$, $j = 1, \ldots, n$, are given. Then the resulting *schedule*, i.e., which task runs a job at which time, repeats itself after the largest period q_k. The smallest period q_1 partitions the time-horizon $[0, q_k)$ into *bins* $[i \cdot q_1, (i+1) \cdot q_1)$; see Figure 1 for an example.

Assume that the offsets a_j, $j = 1, \ldots, n$, determine a feasible single-machine schedule for tasks $\tau_1, \ldots, \tau_n$ where no two jobs collide. By a simple shifting argument, we can assume w.l.o.g. that the first task τ_1 with minimum period $p_1 = q_1$ has offset $a_1 = 0$ (see also Figure 1). Now consider an additional task $\tau = (c, p)$ with maximum period $p = q_k$. Clearly, one can find a feasible offset for this task if and only if one of the q_k/q_1 bins has a free slot of size c.

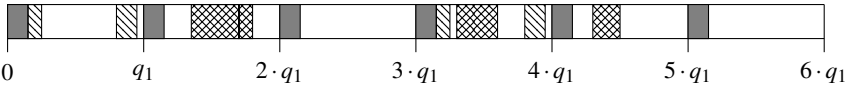


Fig. 1. A schedule for a single machine. The gray jobs belong to tasks with period length q_1, the striped jobs to tasks with period length $q_2 = 3 \cdot q_1$, and the checkered jobs to tasks with period length $q_3 = 6 \cdot q_1$.

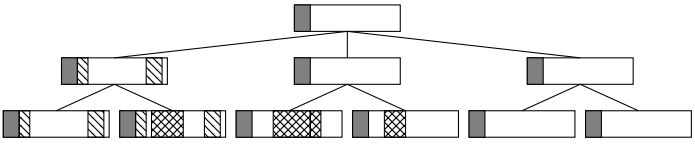


Fig. 2. The full bin tree corresponding to the schedule in Figure 1

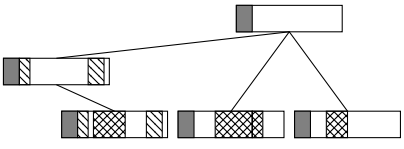


Fig. 3. The compact form of the bin tree in Figure 2

Consider two bins $B_i = [i \cdot q_1, (i+1) \cdot q_1)$ and $B_j = [j \cdot q_1, (j+1) \cdot q_1)$ such that $i \equiv j \mod q_r/q_1$. As far as tasks with period length up to q_r are concerned, these bins look the same. As a shorthand, we write $B_i \equiv_r B_j$ when $i \equiv j \mod q_r/q_1$.

The root of a *full bin tree* is a node representing a bin containing all tasks of period length q_1. It has q_2/q_1 children, each of which represents a bin that contains all its parent's tasks and may contain additional tasks of period length q_2. We say that the root is of period q_1, and its children are of period q_2.

In general, a node B of period q_r contains only tasks of period length up to q_r. If $q_r < q_k$, it has q_{r+1}/q_r children, each of which is a node of period q_{r+1}. Each child of B represents a bin that contains all tasks of B and may contain additional tasks of period length q_{r+1}. Each scheduled task of period length q_r appears in a unique node of period q_r and in all children of that node.

As a consequence of this definition, there is a one-to-one correspondence between nodes of period q_r in the full bin tree and equivalence classes of bins modulo the equivalence relation $\equiv_r$. Furthermore, the hierarchy of equivalence relations $\equiv_r$ $(r = 1, \ldots, k)$ corresponds to the hierarchy of the tree in the following way: If two nodes of period $\geq q_r$ have the same ancestor of period q_r, then their corresponding bins are equivalent modulo $\equiv_r$, and vice versa. In particular the leafs of the bin-tree correspond to the bins of the schedule. Thus we can freely convert between a feasible schedule in terms of task offsets and the corresponding bin tree representation; see Figure 2.

The number of nodes in a full bin tree is dominated by its leaves, of which there are q_k/q_1 many, so we cannot operate efficiently on full bin trees and, in particular, we cannot afford to store a full bin tree to implement our First-Fit heuristic. However, if a node of period q_r does not contain a task of period q_r, it is completely determined by its parent. Therefore, we only need to store those nodes of the tree that introduce a new task to the schedule, see Figure 3 for an example. In this way we have a *compressed bin tree* whose number of nodes is bounded by the number of tasks and that can be constructed in polynomial time.

Coming back to the implementation of First-Fit, given a bin tree and an unscheduled task τ whose period length is greater or equal to the period lengths of all tasks in the bin tree, one can easily determine whether τ can be inserted into the bin tree in compact form by checking all the leaves of the compact tree, as well as all nodes to which a new leaf of the right period can be added. We will use this fact in the next subsection.

2.2 First-Fit Algorithm

In the sequel we use the following term: If a subset of tasks is assigned to a machine, then the *type* of this machine is the smallest period of these tasks. The First-Fit algorithm maintains a list $M_1, \ldots, M_\ell$ of open machines where M_i was opened before M_j if $i < j$. The algorithm is initialized with the empty list. Then, the algorithm proceeds as follows. For $j = 1, \ldots, n$:

1. Find the first machine on which τ_j fits and insert it into a leaf of that machine's bin tree such that no gaps are created within the leaf. Within this machine's bin tree, break ties between leaves arbitrarily.
2. If τ_j does not fit on any open machine, we open a new machine of type p_j and add τ_j to the root node of its bin tree. Furthermore, to simplify the analysis later, we open a *second* new machine of type p_j. On this machine we schedule a dummy task with running time 0 and period p_j.

Note that, because tasks are added in non-decreasing order of period lengths, we only add tasks to leaves of the compressed bin tree as discussed above.

The *utilization* of a task $\tau = (c, p)$ is defined as $\text{util}(\tau) = c/p$. For a set of tasks I and a machine M we define $\text{util}(I) = \sum_{\tau \in I} \text{util}(\tau)$ and $\text{util}(M) = \sum_{\tau \text{ on } M} \text{util}(\tau)$, respectively. The utilization $\text{util}(I)$ is a lower bound for $OPT(I)$. The main result of this section is the following theorem.

Theorem 1. *The First-Fit algorithm is a 2-approximation algorithm.*

Before we present the proof of Theorem 1 we will discuss some differences to the analysis of the First-Fit algorithm for BIN-PACKING and motivate the concept of a *witness* that turns out to be very useful. The simple observation showing that First-Fit for BIN-PACKING is a 2-approximation is as follows: If First-Fit opens a new bin, then let α be the minimum weight of all previously opened bins. This implies that the current item has weight at least $1 - \alpha$ and if there are any other open bins, they must have weight at least $\max\{\alpha, 1 - \alpha\}$. The average weight of the bins is thus at least $1/2$.

Now suppose that the First-Fit algorithm for the machine minimization problem opens a new machine for task $\tau_j = (c_j, p_j)$. If the *type* q of some machine with low utilization is smaller than the running time c_j of the task, then this task cannot be run on this machine. Thus, it may happen that there are many open machines with a low utilization. In particular, it is not true that the average utilization of the open machines is at least $1/2$. However, we can derive a lower bound on the average load of the machines whose type is *compatible* with τ_j, where the set of compatible types is denoted by $Q(j) := \{q \in Q \colon c_j \leq q \leq p_j\}$.

Lemma 1. *Suppose that the First-Fit algorithm cannot schedule τ_j on one of its open machines and opens two new machines instead. Let $q \in Q(j)$ be a compatible machine type, and let $M_1, \ldots, M_\ell$, $\ell > 0$, be the machines of type q that were open before the algorithm tries to assign τ_j. Then, $\frac{1}{\ell}\sum_{i=1}^{\ell} \text{util}(M_i) > \frac{1}{2}$.*

Proof. First observe that $\ell \geq 2$ because the First-Fit algorithm always opens two machines of the same type at a time. Consider the bin trees corresponding to machines $M_1, \ldots, M_\ell$ when τ_j should be added. Note that the leaves of the trees are of period at most p_j. Let $\alpha > 0$ be the minimum fill ratio over all leaf-bins of the trees. If $\alpha > \frac{1}{2}$, then every bin is more than half filled and the claim follows.

Thus, we can assume that $\alpha \leq \frac{1}{2}$. Let $\overline{B}$ be a leaf bin of fill ratio α, and $\overline{M}$ be the machine $\overline{B}$ belongs to. Thus, the utilization of $\overline{M}$ is at least α. We will show that all leaf bins of the machines $\{M_1, \ldots, M_\ell\} \setminus \{\overline{M}\}$ have a fill ratio greater than $1 - \alpha$. This implies, in particular, that all machines other than $\overline{M}$ have a utilization greater than $1 - \alpha$. Since $\ell \geq 2$, the claim then follows by an averaging argument.

Let B be a leaf bin on a machine $M_i \neq \overline{M}$. There are two cases to consider: The First-Fit algorithm considers M_i *before* or *after* $\overline{M}$. If M_i is considered before $\overline{M}$, let τ be any task assigned to $\overline{B}$. Now consider the time when τ was assigned by First-Fit. At that time, either B or an ancestor of B was a leaf-bin. First-Fit tried to assign τ to B (or its ancestor) but failed. Now τ fills at most an α-fraction of a bin, which implies that the fill ratio of B (or its ancestor) must have been more than $1 - \alpha$, otherwise τ would have been packed there instead. Thus, B has fill ratio greater than $1 - \alpha$ at the time τ is assigned. For the case where M_i is considered after $\overline{M}$, we can analogously argue that a task in B could have been assigned to $\overline{B}$ (or an ancestor). □

Let τ_j be a task and let $\mathscr{M}_j$ be the set of machines that are open when First-Fit tries to assign τ_j. Let $Q' \subseteq Q(j)$ be a subset of the compatible machine types and let $\mathscr{M}' \subseteq \mathscr{M}_j$ be the subset of machines whose type is in Q'. The above lemma implies that, if First-Fit opens two new machines when it tries to assign τ_j, then the average load of the machines in $\mathscr{M}'$ is at least $1/2$. We say that τ_j is a *witness* of $\mathscr{M}'$. In particular, if $q < p_j$ is compatible with τ_j and if $\mathscr{M}_q$ denotes the machines of type q that are created by First-Fit, then τ_j is a witness of $\mathscr{M}_q$. We have the following lemma.

Lemma 2. *$FF(I)$ denotes the number of machines opened by the First-Fit heuristic. If for all $q \in Q$, $q < q_k$, the set $\mathscr{M}_q$ has a witness, then $FF(I) \leq 2OPT(I)$.*

Proof. Let $q \in Q$ with $q < q_k$, and let τ be a witness for $\mathscr{M}_q$. Then we can apply Lemma 1 to show that $\sum_{M \in \mathscr{M}_q} \text{util}(M) \geq \frac{1}{2}|\mathscr{M}_q|$. Now let $\mathscr{M}'$ be the set $\mathscr{M}_{q_k}$ without the two last machines that we call $\widetilde{M}_1$ and $\widetilde{M}_2$. Let τ be a task assigned to $\widetilde{M}_1$. Observe that τ is a witness for $\mathscr{M}'$. Thus, $\sum_{M \in \mathscr{M}'} \text{util}(M) \geq \frac{1}{2}|\mathscr{M}'|$. Hence we have $FF(I) - 2 \leq 2 \cdot \text{util}(I \setminus \{\tau\})$ which implies $FF(I) - 2 < 2 \cdot \text{util}(I) \leq 2 \cdot OPT(I)$. Since both $FF(I)$ and $2 \cdot OPT(I)$ are even, we conclude $FF(I) \leq 2 \cdot OPT(I)$. □

If the special case of Lemma 2 does not apply, we can identify sub-instances that are pairwise independent of each other, yet cover all machines opened by First-Fit. We use this observation to prove Theorem 1 by induction.

Proof (of Theorem 1). We prove the theorem by induction on k, the number of different periods. If $k = 1$, then the claim is obvious. Now assume that First-Fit is a 2-approximation for all instances with less than k periods. If for all $q \in Q$, $q < q_k$, the set $\mathscr{M}_q$ has a witness, again the claim follows directly with Lemma 2.

Thus, let $q \in Q$, $q < q_k$ be a period such that $\mathscr{M}_q$ does not have a witness. We now partition the tasks into $I' := \{\tau \in I\colon \text{ First-Fit assigns } \tau \text{ to a machine of type } \leq q\}$ and $I'' := I \backslash I'$. Moreover, let $\overline{I} := \{\tau_i \in I'\colon p_i \leq q\}$. Let τ_j be an arbitrary task in I''. Then q is not compatible with τ_j since otherwise τ_j would be a witness for $\mathscr{M}_q$. Thus, each task in I'' has a running time $> q$. As the period lengths of all tasks in $\overline{I}$ are at most q, no task in $\overline{I}$ can be scheduled together with a task in I''. This shows that $\overline{I}$ and I'' are independent in the sense that $OPT(\overline{I}) + OPT(I'') = OPT(\overline{I} \cup I'') \leq OPT(I)$.

On the other hand, since First-Fit assigns each task of I' to a machine of type $\leq q$ and these must have been opened and typed by a job in $\overline{I}$, one has $FF(\overline{I}) = FF(I')$ and $FF(I) = FF(\overline{I}) + FF(I'')$. Using the induction hypothesis, we get

$$FF(I) = FF(\overline{I}) + FF(I'') \leq 2OPT(\overline{I}) + 2OPT(I'') \leq 2OPT(I).$$

This concludes the proof. □

Surprisingly, the approximation result in Theorem 1 is tight, even if one aims for asymptotic approximation guarantees. This is in sharp contrast to the classical BIN-PACKING problem, where one has a fully polynomial asymptotic approximation scheme [KK82] and for which one does not know variants of First-Fit with optimal (asymptotic) approximation ratios. The proof of the following theorem can be found in [EHN+10]. It is established via a reduction from PARTITION and boosting.

Theorem 2. *Unless $P = NP$, there is no approximation algorithm with a guarantee of $(2-\varepsilon)OPT + o(OPT)$ for the case of harmonic periods, for any $\varepsilon > 0$.*

2.3 An APTAS for a Constant Number of Periods

For the case of harmonic periods the 2-approximation algorithm cannot be improved, unless $P = NP$. This holds even asymptotically. In particular, there is no hope for an asymptotic PTAS. Our experience with real-world instances from our industrial partner, however, is that these instances often have only very few different period lengths. We thus consider the case where the number of different periods k is bounded by a constant and present an asymptotic PTAS for this restricted setting.

Theorem 3. *For any $\varepsilon > 0$ and k bounded by a constant, there is an efficient algorithm that computes a solution of value at most $(1+\varepsilon)OPT(I) + k$.*

We only sketch the main ideas of the algorithm. For a complete description we refer to the full version of this paper [EHN+10]. Although the high level idea is the same as for the APTAS for BIN-PACKING [FdlVL81], the more complex nature of the periodic maintenance problem imposes novel and interesting difficulties. In particular, we cannot simply classify tasks as *big* or *small*, since different bin sizes occur on different machines. Therefore we must do this classification relative to the size of the bin, making the rounding procedure significantly more involved. After the rounding, we determine

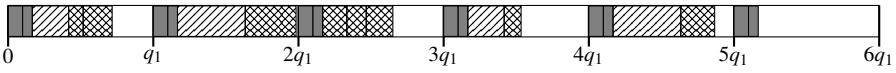


Fig. 4. A schedule in bin structure. The gray jobs belong to tasks with period length q_1, the striped jobs to tasks with period length $q_2 = 3 \cdot q_1$ and the checkered jobs to tasks with period length $q_3 = 6 \cdot q_1$.

the optimal solution for the big tasks by enumeration, making non-trivial use of the concept of bin trees. Finally, we add the small tasks using First-Fit. All these steps have to be performed with extreme care not to introduce extra additive factors to the guarantee on the objective function.

First, we show that the individual schedule for each machine can be assumed to have a particular structure. More precisely, we say that a schedule obeys the *bin structure* if each bin satisfies the following (see also Figure 4): (i) Let q be the smallest period length of a task on the machine; then, there is a task τ_j with $p_j = q$ and $a_j = 0$; (ii) if a task τ_j starts before a task $\tau_{j'}$ in the bin, then $p_j \leq p_{j'}$; and (iii) all jobs in a bin are processed consecutively.

Notice that the same bin tree structure described in Section 2.1 can be used for solutions that follow the bin structure. Moreover, a simple swapping argument shows the following lemma.

Lemma 3. *Let I' be a set of tasks assigned to a machine M. If there is a feasible schedule for I', then there is a feasible schedule that obeys the bin structure.*

The structure given by this lemma is crucial to describe schedules compactly: Given an assignment of tasks to bins in the bin tree such that no bin is overloaded, we can find an offset for each task such that no two tasks collide. In particular, in the enumeration step of our APTAS, we will only enumerate over the assignments of the tasks to the bins, not the individual schedules within each bin.

Rounding. In order to round the execution times of big tasks, more forethought is necessary compared to the APTAS for BIN-PACKING. We want to achieve two goals: On the one hand, only a constant number of big tasks should fit into a bin. On the other hand, we want to ensure that, later on, the remaining space in the bins can be used efficiently by the First-Fit procedure for the small tasks.

Given a bin size $q \in Q$, we say that a task τ_j is *small* with respect to q if $c_j < \varepsilon q$ and *big* if $\varepsilon q \leq c_j \leq q$. Now the rounding is done independently for each period length q_ℓ: For $i = 1, \ldots, k$, in iteration i, we only round tasks that are big with respect to q_i, and that have not been rounded before. These tasks are sorted by non-decreasing execution times and then partitioned into $O(1/\varepsilon^2)$ many groups. All groups are of equal size except for the first group which may have fewer elements. The execution times of all tasks of the same group are then rounded up to the maximum execution time of that group.

Denote by I' the rounded instance. For each period length q_ℓ and each bin size q_i, denote by $L_{\ell,i}$ the set of tasks on the last group. We define $L := \cup_{\ell,i} L_{\ell,i}$ and $J := I' \setminus L$. Using $OPT(I)$ as a template for J, one can show that $OPT(J) \leq OPT(I)$. For each bin

size q, denote by C_q the set of all execution times c of big tasks in J such that $c \leq q$ (and thus a task with execution time c could possibly be scheduled in such a bin). The rounding ensures that the cardinality of each set C_q is bounded by a constant.

We will show at the end that the tasks in L together with a certain subset of the small tasks can be packed onto $O(\varepsilon)OPT(I)+k$ extra machines. Now, we proceed with presenting an approximation algorithm for J.

Enumeration of Solutions. In order to enumerate the solutions of big tasks in J, it is crucial to compactly represent the bin trees. We introduce the concept of a *bin configuration* which, given a bin schedule, encodes: (a) the level of a bin on the bin tree; (b) the size of the bin q; (c) the assignment of big tasks; and (d) the space reserved for small tasks for each period length, rounded down to the nearest multiple of εq.

Since the number of big tasks in a bin is at most $1/\varepsilon$, and $|C_q|$ is bounded by a constant, the total number of bin configurations is also bounded by a constant. We can therefore enumerate the bin configurations on the compressed bin trees in polynomial time as follows:

(i) Assume that, for each machine, we have already determined the bin configurations of all bins up to level q_i in its corresponding compressed bin trees.
(ii) For each bin configuration of level q_{i+1}, we enumerate the number of bins following this configuration. In what follows, we consider only the iteration in which we enumerate the respective amount that corresponds to the optimal solution to J.
(iii) Attach each node configuration to a parent node on any machine, such that the jobs of period lengths up to q_i coincide.

One of the enumerated solutions corresponds—up to permutation of sub-trees—to an optimal solution to J. Indeed, by an inductive argument, we can assume that levels up to q_i contains the right number of nodes of each configuration. Then, if in Step (ii) we enumerate the correct number of bins with each configuration, it is clear that Step (iii) will successfully attach the nodes to the trees.

Moreover, the running time of the algorithm is polynomial. Indeed, since the number of configurations on each level is at most a constant λ, Step (ii) can be performed in time $O(n^{\lambda})$. Thus, the running time of the whole algorithm is bounded by $O(n^{k\lambda})$.

The APTAS. We are now ready to state the APTAS: (1) Round tasks and define instances J and L; (2) enumerate over all possible configurations of bin trees; (3) assign the big tasks in J according to the bin configurations; (4) greedily assign small tasks to bins by using the reserved space in the configurations; call S the set of small tasks that could not be assigned; (5) for each period length q independently, schedule all jobs in $L \cup S$ of period q with the classical First-Fit algorithm for BIN-PACKING. (Notice that, in this last step, we do not mix tasks of different period lengths on the same machine.)

Up to Step (4) of the algorithm, we have only used $OPT(J) \leq OPT(I)$ many machines. It remains to show that First-Fit in Step (5) assigns tasks in $L \cup S$ to at most $O(\varepsilon)OPT + k$ machines. We omit the proof and refer to [EHN+10].

Lemma 4. *First-Fit assigns tasks in $L \cup S$ to at most $O(\varepsilon)OPT + k$ machines.*

If not only k is bounded by a constant but even q_k/q_1, one can obtain a better bound.

Theorem 4. *For any $\varepsilon > 0$ and q_k/q_1 bounded by a constant, there is an efficient algorithm that computes a solution of value at most $(1+\varepsilon)OPT(I)+1$.*

The main difference to the algorithm above is that here a task τ_i is small if $c_i \leq \varepsilon q_1$ and big otherwise. Also, we need to be more careful with the First-Fit procedure to assign the remaining small jobs as in Step (5). For full details we refer to [EHN+10].

3 Arbitrary Period Lengths

In this section we study the periodic maintenance problem in the setting of arbitrary, i.e., not necessarily harmonic, period lengths. We only sketch the results. For more details see [EHN+10].

One can derive a First-Fit algorithm as follows: Partition the tasks according to their period lengths. Then do First-Fit for period lengths separately such that no two jobs with different period lengths are scheduled on the same machine. Denote by $FF(I)$ the number of machines in the resulting schedule. Using utilization bound techniques similar to BIN-PACKING, we obtain the following theorem.

Theorem 5. *For any instance I it holds that $FF(I) \leq 2 \cdot OPT + k - 1$.*

Now assume that k is bounded by some constant and let $\varepsilon > 0$. We call a task τ *small* if $\text{util}(\tau) \leq \varepsilon$, otherwise we call τ *big*. We define the sets I_{small} and I_{big} respectively. We enumerate all solutions for the big tasks and call the best solution $ENUM(I_{big})$. This can be done in polynomial time since there are at most $\lfloor 1/\varepsilon \rfloor$ big tasks on each machine and k is assumed to be constant (a similar argument as used in [FdlVL81] for BIN-PACKING). We output $EFF(I) := \min\{FF(I), ENUM(I_{big}) + FF(I_{small})\}$.

Theorem 6. *Assume that k is bounded by a constant. Then, $EFF(I)$ can be computed in polynomial time and $EFF(I) \leq (3/2 + O(\varepsilon))\,OPT + k$.*

In the remainder of this section we show that the periodic maintenance minimization problem is hard to approximate within a factor of $n^{1-\varepsilon}$ using an approximation preserving reduction from COLORING. We remark that this reduction has been stated independently in [BBNS02, Bha98]. Again, we only sketch the results and refer to [EHN+10] for more details. The result still holds if we restrict the problem to unit execution times.

For unit execution times, a set of offsets is feasible if and only if $a_i + k_i \cdot p_i \neq a_j + k_j \cdot p_j$ for all i, j and $k_i, k_j \in \mathbb{N}_0$. With elementary number theory [NZM91] it is easy to see that this is equivalent to $a_i \not\equiv a_j \bmod \gcd(p_i, p_j)$. The reduction works as follows. Let the graph $G = (V, E)$ be an instance of the COLORING problem. Let $\overline{E}$ be the complement of E, i.e., $\overline{E} := \{\{u, v\} : u, v \in V, \{u, v\} \notin E\}$. We choose pairwise different primes q_e for all $e \in \overline{E}$. This can be done in polynomial time with the sieve of Eratosthenes since the Prime Number Theorem guarantees $\Theta(x/\ln(x))$ primes among the first x natural numbers (see, e.g., [NZM91]). For each node $v \in V$, we define a task $\tau_v = (c_v, p_v)$ with $c_v := 1$ and $p_v := \prod_{e \in \overline{E}:\, v \in e} q_e$. We denote with $\text{red}(G) := \{\tau_v : v \in V\}$ the periodic maintenance instance obtained from the graph G using this construction. Note that we can compute $\text{red}(G)$ in polynomial time. Our construction has the following properties:

Lemma 5. *Given a graph $G = (V,E)$, the machine minimization instance* $\mathrm{red}(G)$ *satisfies the following properties.*

a) For each edge $\{u,v\} \in E$, tasks τ_u and τ_v cannot be assigned to the same machine.
b) For an independent set $U \subseteq V$, tasks $\{\tau_v : v \in U\}$ can be scheduled on one machine.

These properties can be used to show that our construction is a reduction.

Lemma 6. *For each $k \in \mathbb{N}$, a graph G is k-colorable if and only if* $\mathrm{red}(G)$ *can be scheduled on k machines.*

It is shown in [Zuc07] that it is NP-hard to approximate the COLORING problem within a factor of $n^{1-\varepsilon}$, for any constant $\varepsilon > 0$, where n is the number of nodes of the graph. This immediately implies the claimed inapproximability result.

Theorem 7. *The periodic maintenance minimization problem cannot be approximated within a factor of $n^{1-\varepsilon}$, for any constant $\varepsilon > 0$, unless $P = NP$.*

We remark that the primes generated in the reduction can get exponentially large (although their encoding length is polynomial). For that reason, this result only shows that it is weakly NP-hard to approximate within the factor given in Theorem 7. The question whether there is a pseudopolynomial time algorithm with a constant factor approximation guarantee remains open.

References

[BBNS02] Bar-Noy, A., Bhatia, R., Naor, J., Schieber, B.: Minimizing service and operation costs of periodic scheduling. Math. Oper. Res. 27(3) (2002)

[Bha98] Bhatia, R.: Approximation Algorithms for Scheduling Problems. PhD thesis, University of Maryland (1998)

[BHR93] Baruah, S.K., Howell, R.R., Rosier, L.E.: Feasibility problems for recurring tasks on one processor. In: Selected papers of the 15th International Symposium on Mathematical Foundations of Computer Science, pp. 3–20. Elsevier, Amsterdam (1993)

[BRTV90] Baruah, S., Rousier, L., Tulchinsky, I., Varvel, D.: The complexity of periodic maintenance. In: Proceedings of the International Computer Symposium (1990)

[But04] Buttazzo, G.C.: Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications. Springer, Heidelberg (2004)

[CSW08] Conforti, M., Di Summa, M., Wolsey, L.A.: The mixing set with divisible capacities. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO 2008. LNCS, vol. 5035, pp. 435–449. Springer, Heidelberg (2008)

[EHN+10] Eisenbrand, F., Hähnle, N., Niemeier, M., Skutella, M., Verschae, J., Wiese, A.: Scheduling periodic tasks in a hard real-time environment. Technical report, EPF Lausanne & TU Berlin (February 2010), http://disopt.epfl.ch/webdav/site/disopt/shared/PM_EHNSVW10_report.pdf

[FdlVL81] de la Fernandez Vega, W., Lueker, G.S.: Bin packing can be solved within $1+\varepsilon$ in linear time. Combinatorica 1, 349–355 (1981)

[GJ79] Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-Completeness. Freemann, New York (1979)

[KAL96] Korst, J., Aarts, E., Lenstra, J.K.: Scheduling periodic tasks. INFORMS Journal on Computing 8, 428–435 (1996)

[KALW91] Korst, J., Aarts, E., Lenstra, J.K., Wessels, J.: Periodic multiprocessor scheduling. In: Aarts, E.H.L., van Leeuwen, J., Rem, M. (eds.) PARLE 1991. LNCS, vol. 505, pp. 166–178. Springer, Heidelberg (1991)

[KK82] Karmakar, N., Karp, R.M.: An efficient approximation scheme for the one-dimensional binpacking problem. In: Foundations of Computer Science (FOCS), vol. 23, pp. 312–320 (1982)

[Leu04] Leung, J.Y.-T.: Handbook of Scheduling: Algorithms, Models and Performance Analysis. Chapman & Hall/CRC (2004)

[Mar85] Marcotte, O.: The cutting stock problem and integer rounding. Mathematical Programming 33, 82–92 (1985)

[NZM91] Niven, I., Zuckerman, H.S., Montgomery, H.L.: An Introduction to the Theory of Numbers, 5th edn. Wiley, Chichester (1991)

[SL94] Simchi-Levi, D.: New worst-case results for the bin-packing problem. Naval Research Logistics 41, 579–585 (1994)

[VA97] Verhaegh, W.F.J., Aarts, E.H.L.: A polynomial-time algorithm for knapsack with divisible item sizes. Information Processing Letters 62, 217–221 (1997)

[WL83] Wei, W.D., Liu, C.L.: On a periodic maintenance problem. Operations Research Letters 2, 90–93 (1983)

[ZIRdF08] Zhao, M., de Farias Jr., I.R.: The mixing-MIR set with divisible capacities. Mathematical Programming 115, 73–103 (2008)

[Zuc07] Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. Theory of Computing 3, 103–128 (2007)

Scalably Scheduling Power-Heterogeneous Processors

Anupam Gupta[1,⋆], Ravishankar Krishnaswamy[1,⋆], and Kirk Pruhs[2,⋆⋆]

[1] Computer Science Dept., Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2] Computer Science Dept., University of Pittsburgh, Pittsburgh, PA 15260, USA

Abstract. We show that a natural online algorithm for scheduling jobs on a heterogeneous multiprocessor, with arbitrary power functions, is scalable for the objective function of weighted flow plus energy.

1 Introduction

Many prominent computer architects believe that architectures consisting of heterogeneous processors/cores, such as the STI Cell processor, will be the dominant architectural design in the future [8,13,12,17,18]. The main advantage of a heterogeneous architecture, relative to an architecture of identical processors, is that it allows for the inclusion of processors whose design is specialized for particular types of jobs, and for jobs to be assigned to a processor best suited for that job. Most notably, it is envisioned that these heterogeneous architectures will consist of a small number of high-power high-performance processors for critical jobs, and a larger number of lower-power lower-performance processors for less critical jobs. Naturally, the lower-power processors would be more energy efficient in terms of the computation performed per unit of energy expended, and would generate less heat per unit of computation. For a given area and power budget, heterogeneous designs can give significantly better performance for standard workloads [8,17]; Emulations in [12] suggest a figure of 40% better performance, and emulations in [18] suggest a figure of 67% better performance. Moreover, even processors that were designed to be homogeneous, are increasingly likely to be heterogeneous at run time [8]: the dominant underlying cause is the increasing variability in the fabrication process as the feature size is scaled down (although run time faults will also play a role). Since manufacturing yields would be unacceptably low if every processor/core was required to be perfect, and since there would be significant performance loss from derating the entire chip to the functioning of the least functional processor (which is what would be required in order to attain processor homogeneity), some processor heterogeneity seems inevitable in chips with many processors/cores.

⋆ Supported in part by NSF award CCF-0729022 and an Alfred P. Sloan Fellowship.
⋆⋆ Supported in part by NSF grants CNS-0325353, IIS-0534531, and CCF-0830558, and an IBM Faculty Award.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 312–323, 2010.

The position paper [8] identifies three fundamental challenges in scheduling heterogeneous multiprocessors: (1) the OS must discover the status of each processor, (2) the OS must discover the resource demand of each job, and (3) given this information about processors and jobs, the OS must match jobs to processors as well as possible. In this paper, we address this third fundamental challenge. In particular, we assume that different jobs are of differing importance, and we study how to assign these jobs to processors of varying power and varying energy efficiency, so as to achieve the best possible trade-off between energy and performance.

Formally, we assume that a collection of jobs arrive in an online fashion over time. When a job j arrives in the system, the system is able to discover a *size* $p_j \in \mathbb{R}_{>0}$, as well as a *importance/weight* $w_j \in \mathbb{R}_{>0}$, for that job. The importance w_j specifies an upper bound on the amount of energy that the system is allowed to invest in running j to reduce j's flow by one unit of time (assuming that this energy investment in j doesn't decrease the flow of other jobs)—hence jobs with high weight are more important, since higher investments of energy are permissible to justify a fixed reduction in flow. Furthermore, we assume that the system knows the allowable speeds for each processor, and the system also knows the power used when each processor is run at its set of allowable speeds. We make no real restrictions on the allowable speeds, or on the power used for these speeds.[1] The online scheduler has three component policies:

Job Selection: Determines which job to run on each processor at any time.
Speed Scaling: Determines the speed of each processor at each time.
Assignment: When a new job arrives, it determines the processor to which this new job is assigned.

The objective we consider is that of *weighted flow plus energy*. The rationale for this objective function is that the optimal schedule under this objective gives the best possible weighted flow for the energy invested, and increasing the energy investment will not lead to a corresponding reduction in weighted flow (intuitively, it is not possible to speed up a collection of jobs with an investment of energy proportional to these jobs' importance).

We consider the following natural online algorithm that essentially adopts the job selection and speed scaling algorithms from the uniprocessor algorithm in [5], and then greedily assigns the jobs based on these policies.

Job Selection: Highest Density First (HDF)
Speed Scaling: The speed is set so that the power is the fractional weight of the unfinished jobs.
Assignment: A new job is assigned to the processor that results in the least increase in the projected future weighted flow, assuming the adopted speed scaling and job selection policies, and ignoring the possibility of jobs arriving in the future.

[1] So the processors may or may not be speed scalable, the speeds may be continuous or discrete or a mixture, the static power may or may not be negligible, the dynamic power may or may not satisfy the cube root rule, etc.

Our main result is then:

Theorem 1. *This online algorithm is scalable for scheduling jobs on a heterogeneous multiprocessor with arbitrary power functions to minimize the objective function of weighted flow plus energy.*

In this context, *scalable* means that if the adversary can run processor i at speed s and power $P(s)$, the online algorithm is allowed to run the processor at speed $(1+\epsilon)s$ and power $P(s)$, and then for all inputs, the online cost is bounded by $O(f(\epsilon))$ times the optimal cost. Intuitively, a scalable algorithm can handle almost the same load as optimal; for further elaboration, see [20,19]. Theorem 1 extends theorems showing similar results for weighted flow plus energy on a uniprocessor [5,2], and for weighted flow on a multiprocessor without power considerations [9]. As scheduling on identical processors with the objective of total flow, and scheduling on a uniprocessor with the objective of weighted flow, are special cases of our problem, constant competitiveness is not possible without some sort of resource augmentation [16,3].

Our analysis is an amortized local-competitiveness argument. As is usually the case with such arguments, the main technical hurdle is to discover the "right" potential function. The most natural strawman potential function to try is the sum over all processors of the single processor potential function used in [5]. While one can prove constant competitiveness with this potential in some special cases (e.g. where for each processor the allowable speeds are the non-negative reals, and the power satisfies the cube-root rule), one can not prove constant competitiveness for general power functions with this potential function. The reason for this is that the uniprocessor potential function from [5] is not sufficiently accurate. Specifically, one can construct configurations where the adversary has finished all jobs, and where the potential is much higher than the remaining online cost. This did not mess up the analysis in [5] because to finish all these jobs by this time the adversary would have had to run very fast in the past, wasting a lot of energy, which could then be used to pay for this unnecessarily high potential. But since we consider multiple processors, the adversary may have no jobs left on a particular processor simply because it assigned these jobs to a different processor, and there may not be a corresponding unnecessarily high adversarial cost that can be used to pay for this unnecessarily high potential.

Thus, the main technical contribution in this paper is a seemingly more accurate potential function expressing the additional cost required to finish one collection of jobs compared to another collection of jobs. Our potential function is arguably more transparent than the one used in [5], and we expect that this potential function will find future application in the analysis of other power management algorithms.

In section 3, we show that a similar online algorithm is $O(1/\epsilon)$-competitive with $(1+\epsilon)$-speedup for *unweighted* flow plus energy. We also remark that when the power functions $P_i(s)$ are restricted to be of the form s^{α_i}, our algorithms give a $O(\alpha^2)$-competitive algorithm (with no resource augmentation needed) for the problem of minimizing weighted flow plus energy, and an $O(\alpha)$-competitive algorithm for minimizing the unweighted flow plus energy, where $\alpha = \max_i \alpha_i$.

1.1 Related Results

Let us first consider previous work for the case of a single processor, with unbounded speed, and a polynomially bounded power function $P(s) = s^\alpha$. [21] gave an efficient offline algorithm to find the schedule that minimizes average flow subject to a constraint on the amount of energy used, in the case that jobs have unit work. [1] introduced the objective of flow plus energy and gave a constant competitive algorithm for this objective in the case of unit work jobs. [6] gave a constant competitive algorithm for the objective of weighted flow plus energy. The competitive ratio was improved by [15] for the unweighted case using a potential function specifically tailored to integer flow. [4] extended the results of [6] to the bounded speed model, and [10] gave a *nonclairvoyant* algorithm that is $O(1)$-competitive.

Still for a single processor, dropping the assumptions of unbounded speed and polynomially-bounded power functions, [5] gave a 3-competitive algorithm for the objective of unweighted flow plus energy, and a 2-competitive algorithm for fractional weighted flow plus energy, both in the uniprocessor case for a large class of power functions. The former analysis was subsequently improved by [2] to show 2-competitiveness, along with a matching lower bound on the competitive ratio.

Now for multiple processors: [14] considered the setting of multiple *homogeneous* processors, where the allowable speeds range between zero and some upper bound, and the power function is polynomial in this range. They gave an algorithm that uses a variant of round-robin for the assignment policy, and job selection and speed scaling policies from [6], and showed that this algorithm is scalable for the objective of (unweighted) flow plus energy. Subsequently, [11] showed that a randomized machine selection algorithm is scalable for weighted flow plus energy (and even more general objective functions) in the setting of polynomial power functions. Both these algorithms provide non-migratory schedules and compare their costs with optimal solutions which could even be migratory. In comparison, as mentioned above, for the case of polynomial power functions, our techniques can give a deterministic constant-competitive online algorithm for non-migratory weighted flow time plus energy. (Details appear in the final version.)

In non-power-aware settings, the paper most relevant to this work is that of [9], which gives a scalable online algorithm for minimizing weighted flow on unrelated processors. Their setting is even more demanding, since they allow the processing requirement of the job to be processor dependent (which captures a type of heterogeneity that is orthogonal to the performance energy-efficiency heterogeneity that we consider in this paper). Our algorithm is based on the same general intuition as theirs: they assign each new job to the processor that would result in the least increment in future weighted flow (assuming HDF is used for job selection), and show that this online algorithm is scalable using an amortized local competitiveness argument. However, it is unclear how to directly extend their potential function to our power-aware setting; we had success only in the case that each processor had allowable speed-power combinations lying in $\{(0,0),(s_i,P_i)\}$.

1.2 Preliminaries

Scheduling Basics. We consider only non-migratory schedules, which means that no job can ever run on one processor, and later run on some other processor. In general, migration is undesirable as the overhead can be significant. We assume that preemption is allowed, that is, that jobs may be suspended, and restarted later from the point of suspension. It is clear that if preemption is not allowed, bounded competitiveness is not obtainable. The speed is the rate at which work is completed; a job j with size p_j run at a constant speed s completes in $\frac{p_j}{s}$ seconds. A job is completed when all of its work has been processed. The flow of a job is the completion time of the job minus the release time of the job. The weighted flow of a job is the weight of the job times the flow of the job. For a $t \geq r_j$, let $p_j(t)$ be the remaining unprocessed work on job j at time t. The fractional weight of job j at this time is $w_j \frac{p_j(t)}{p_j}$. The fractional weighted flow of a job is the integral over times between the job's release time and its completion time of its fractional weight at that time. The density of a job is its weight divided by its size. The job selection policy Highest Density First (HDF) always runs the job of highest density. The inverse density of a job is its size divided by its weight.

Power Functions. The power function for processor i is denoted by $P_i(s)$, and specifies the power used when processor is run at speed s. We essentially allow any reasonable power function. However, we do require the following minimal conditions on each power function, which we adopt from [5]. We assume that the allowable speeds are a countable collection of disjoint subintervals of $[0, \infty)$. We assume that all the intervals, except possibly the rightmost interval, are closed on both ends. The rightmost interval may be open on the right if the power $P_i(s)$ approaches infinity as the speed s approaches the rightmost endpoint of that interval. We assume that P_i is non-negative, and P_i is continuous and differentiable on all but countably many points. We assume that either there is a maximum allowable speed T, or that the limit inferior of $P_i(s)/s$ as s approaches infinity is not zero (if this condition doesn't hold then, then the optimal speed scaling policy is to run at infinite speed). Using transformations specified in [5], we may assume without loss of generality that the power functions satisfy the following properties: P is continuous and differentiable, $P(0) = 0$, P is strictly increasing, P is strictly convex, and P is unbounded. We use Q_i to denote P_i^{-1}; i.e., $Q_i(y)$ gives us the speed that we can run processor i at, if we specify a limit of y.

Local Competitiveness and Potential Functions. Finally, let us quickly review amortized local competitiveness analysis on a single processor. Consider an objective G. Let $G_A(t)$ be the increase in the objective in the schedule for algorithm A at time t. So when G is fractional weighted flow plus energy, $G_A(t)$ is $P_A^t + w_A^t$, where P_A^t is the power for A at time t and w_A^t is the fractional weight of the unfinished jobs for A at time t. Let OPT be the offline adversary that optimizes G. A is locally c-competitive if for all times t, if $G_A(t) \leq c \cdot G_{OPT}(t)$. To prove A is $(c+d)$-competitive using an amortized local competitiveness argument,

it suffices to give a potential function $\Phi(t)$ such that the following conditions hold (see for example [19]).

Boundary condition: Φ is zero before any job is released and Φ is non-negative after all jobs are finished.

Completion condition: Φ does not increase due to completions by either A or OPT.

Arrival condition: Φ does not increase more than $d \cdot OPT$ due to job arrivals.

Running condition: At any time t when no job arrives or is completed,

$$G_A(t) + \frac{d\Phi(t)}{dt} \leq c \cdot G_{OPT}(t) \tag{1}$$

The sufficiency of these conditions for proving $(c + d)$-competitiveness follows from integrating them over time.

2 Weighted Flow

Our goal in this section is to prove Theorem 1. We first show that the online algorithm is $(1 + \epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive for the objective of *fractional* weighted flow plus energy. Theorem 1 then follows since HDF is $(1 + \epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive for fixed processor speeds [7] for the objective of (integer) weighted flow.

Let OPT be some optimal schedule minimizing fractional weighted flow. Let $w^t_{a,i}(q)$ denote the total fractional weight of jobs in processor i's queue that have an inverse density of at least q. Let $w^t_{a,i} := w^t_{a,i}(0)$ be the total fractional weight of unfinished jobs in the queue. Let $w^t_a := \sum_i w^t_{a,i}$ be the total fractional weight of unfinished jobs in all queues. Let $w^t_{o,i}(q)$, $w^t_{o,i}$, and w^t_o be similarly defined for OPT. When the time instant being considered is clear, we drop the superscript of t from all variables.

We assume that once OPT has assigned a job to some processor, it runs the BCP algorithm [5] for job selection and speed scaling—i.e., it sets the speed of the i^{th} processor to $Q_i(w_{o,i})$, and hence the i^{th} processor uses power $W_{o,i}$, and uses HDF for job selection. We can make such an assumption because the results of [5] show that the fractional weighted flow plus energy of the schedule output by this algorithm is within a factor of two of optimal. Therefore, the only real difference between OPT and the online algorithm is the assignment policy.

2.1 The Assignment Policy

To better understand the online algorithm's assignment policy, define the "shadow potential" for processor i at time t to be

$$\widehat{\Phi}_{a,i}(t) = \int_{q=0}^{\infty} \int_{x=0}^{w^t_{a,i}(q)} \frac{x}{Q_i(x)}\, dx\, dq \tag{2}$$

The shadow potential captures (up to a constant factor) the total fractional weighted flow to serve the current set of jobs if no jobs arrive in the future.

Based on this, the online algorithm's assignment policy can alternatively be described as follows:

Assignment Policy. When a new job with size p_j and weight w_j arrives at time t, the assignment policy assigns it to a processor which would cause the smallest increase in the shadow potential; i.e. a processor minimizing

$$\int_{q=0}^{d_j} \int_{x=0}^{w^t_{a,i}(q)+w_j} \frac{x}{Q_i(x)}\,dx\,dq - \int_{q=0}^{d_j} \int_{x=0}^{w^t_{a,i}(q)} \frac{x}{Q_i(x)}\,dx\,dq$$
$$= \int_{q=0}^{d_j} \int_{x=w^t_{a,i}(q)}^{w^t_{a,i}(q)+w_j} \frac{x}{Q_i(x)}\,dx\,dq$$

2.2 Amortized Local Competitiveness Analysis

We apply a local competitiveness argument as described in subsection 1.2. Because the online algorithm is using the BCP algorithm on each processor, the power for the online algorithm is $\sum_i P_i(Q_i(w_{a,i})) = w_a$. Thus $G_A = 2w_a$. Similarly, since OPT is using BCP on each processor $G_{\text{OPT}} = 2w_o$.

Defining the potential function. For processor i, define the potential

$$\Phi_i(t) = \frac{2}{\epsilon} \int_{q=0}^{\infty} \int_{x=0}^{(w^t_{a,i}(q)-w^t_{o,i}(q))_+} \frac{x}{Q_i(x)}\,dx\,dq \tag{3}$$

Here $(\cdot)_+ = \max(\cdot, 0)$. The global potential is then defined to be $\Phi(t) = \sum_i \Phi_i(t)$. Firstly, we observe that the function $x/Q_i(x)$ is increasing and subadditive. Then, the following lemma will be useful subsequently, the proof of which will appear in the full version of the paper.

Lemma 1. *Let g be any increasing subadditive function with $g(0) \geq 0$, and $w_a, w_o, w_j \in \mathbb{R}_{\geq 0}$. Then,*

$$\int_{x=w_a}^{w_a+w_j} g(x)\,dx - \int_{x=(w_a-w_o-w_j)_+}^{(w_a-w_o)_+} g(x)\,dx \leq 2\int_{x=0}^{w_j} g(w_o+x)\,dx$$

That the boundary and completion conditions are satisfied are obvious. In Lemma 2 we prove that the arrival condition holds, and in Lemma 3 we prove that the running condition holds.

Lemma 2. *The arrival condition holds with $d = \frac{4}{\epsilon}$.*

Proof. Consider a new job j with processing time p_j, weight w_j and inverse density $d_j = p_j/w_j$, which the algorithm assigns to processor 1 while the optimal solution assigns it to processor 2. Observe that $\int_{q=0}^{d_j} \int_{x=w_{o,2}(q)}^{w_{o,2}(q)+w_j} \frac{x}{Q_2(x)}\,dx\,dq$ is the increase in OPT's fractional weighted flow due to this new job j. Thus our goal

is to prove that the increase in the potential due to job j's arrival is at most this amount. The change in the potential $\Delta\Phi$ is:

$$\frac{2}{\epsilon}\int_{q=0}^{d_j}\left(\int_{x=(w_{a,1}(q)-w_{o,1}(q))_+}^{(w_{a,1}(q)-w_{o,1}(q)+w_j)_+}\frac{x}{Q_1(x)}\,dx-\int_{x=(w_{a,2}(q)-w_{o,2}(q)-w_j)_+}^{(w_{a,2}(q)-w_{o,2}(q))_+}\frac{x}{Q_2(x)}\,dx\right)dq$$

Now, since $x/Q_1(x)$ is an increasing function we have that

$$\int_{x=(w_{a,1}(q)-w_{o,1}(q))_+}^{(w_{a,1}(q)-w_{o,1}(q)+w_j)_+}\frac{x}{Q_1(x)}\,dx\le\int_{x=w_{a,1}(q)}^{w_{a,1}(q)+w_j}\frac{x}{Q_1(x)}\,dx$$

and hence the change of potential can be bounded by

$$\frac{2}{\epsilon}\int_{q=0}^{d_j}\left(\int_{x=w_{a,1}(q)}^{w_{a,1}(q)+w_j}\frac{x}{Q_1(x)}\,dx-\int_{x=(w_{a,2}(q)-w_{o,2}(q)-w_j)_+}^{(w_{a,2}(q)-w_{o,2}(q))_+}\frac{x}{Q_2(x)}\,dx\right)dq$$

Since we assigned the job to processor 1, we know that

$$\int_{q=0}^{d_j}\int_{x=w_{a,1}(q)}^{w_{a,1}(q)+w_j}\frac{x}{Q_1(x)}\,dx\,dq\le\int_{q=0}^{d_j}\int_{x=w_{a,2}(q)}^{w_{a,2}(q)+w_j}\frac{x}{Q_2(x)}\,dx\,dq$$

Therefore, the change in potential is at most

$$\Delta\Phi\le\frac{2}{\epsilon}\int_{q=0}^{d_j}\left(\int_{x=w_{a,2}(q)}^{w_{a,2}(q)+w_j}\frac{x}{Q_2(x)}\,dx-\int_{x=(w_{a,2}(q)-w_{o,2}(q)-w_j)_+}^{(w_{a,2}(q)-w_{o,2}(q))_+}\frac{x}{Q_2(x)}\,dx\right)dq$$

Applying Lemma 1, we get:

$$\Delta\Phi\le\left(2\cdot\frac{2}{\epsilon}\right)\int_{q=0}^{d_j}\int_{x=w_{o,2}(q)}^{w_{o,2}(q)+w_j}\frac{x}{Q_2(x)}\,dx\,dq$$

Lemma 3. *The running condition holds with constant* $c=1+\frac{1}{\epsilon}$.

Proof. Let us consider an infinitesimally small interval $[t,t+dt)$ during which no jobs arrive and analyze the change in the potential $\Phi(t)$. Since $\Phi(t)=\sum_i\Phi_i(t)$, we can do this on a per-processor basis. Fix a single processor i, and time t. Let $w_i(q):=(w_{a,i}(q)-w_{o,i}(q))_+$, and $w_i:=(w_{a,i}-w_{o,i})_+$. Let q_a and q_o denote the inverse densities of the jobs being executed on processor i by the algorithm and optimal solution respectively (which are the densest jobs in their respective queues, since both run HDF). Define $s_a=Q_i(w_{a,i})$ and $s_o=Q_i(w_{o,i})$. Since we assumed that OPT uses the BCP algorithm on each processor, OPT runs processor i at speed s_o. Since the online algorithm is also using BCP, but has $(1+\epsilon)$-speed augmentation, the online algorithms runs the processor at speed $(1+\epsilon)s_a$. Hence the fractional weight of the job the online algorithm works on decreases at a rate of $s_a(1+\epsilon)/q_a$. Therefore, the quantity $w_{a,i}(q)$ drops by $s_a\,dt(1+\epsilon)/q_a$ for $q\in[0,q_a]$. Likewise, $w_{o,i}(q)$ drops by $s_o\,dt/q_o$ for $q\in[0,q_o]$

due to the optimal algorithm working on its densest job. We consider several different cases based on the values of $q_o, q_a, w_{o,i}$, and $w_{a,i}$ and establish bounds on $d\Phi_i(t)/dt$; Recall the definition of $\Phi_i(t)$ from equation (3):

$$\Phi_i(t) = \frac{2}{\epsilon} \int_{q=0}^{\infty} \int_{x=0}^{(w^t_{a,i}(q) - w^t_{o,i}(q))+} \frac{x}{Q_i(x)}\, dx\, dq$$

Case (1): $w_{a,i} < w_{o,i}$: The only possible increase in potential function occurs due to the decrease in $w_{o,i}(q)$, which happens for values of $q \in [0, q_o]$. But for q's in this range, $w_{a,i}(q) \leq w_{a,i}$ and $w_{o,i}(q) = w_{o,i}$. Thus the inner integral is empty, resulting in no increase in potential. The running condition then holds since $w_{a,i} < w_{o,i}$.

Case (2): $w_{a,i} > w_{o,i}$: To quantify the change in potential due to the online algorithm working, observe that for any $q \in [0, q_a]$, the inner integral of Φ_i decreases by

$$\int_{x=0}^{w_i(q)} \frac{x}{Q_i(x)}\, dx - \int_{x=0}^{w_i(q) - (1+\epsilon)\frac{s_a\, dt}{q_a}} \frac{x}{Q_i(x)}\, dx = \frac{w_i(q)}{Q_i(w_i(q))}(1+\epsilon)\frac{s_a\, dt}{q_a}$$

Here, we have used the fact that dt is infinitisemally small to get the above equality. Hence, the total drop in Φ_i due to the online algorithm's processing is

$$\begin{aligned}\frac{2}{\epsilon} \int_{q=0}^{q_a} \frac{w_i(q)}{Q_i(w_i(q))}(1+\epsilon)\frac{s_a\, dt}{q_a}\, dq &\geq \frac{2}{\epsilon} \int_{q=0}^{q_a} \frac{w_i}{Q_i(w_i)}(1+\epsilon)\frac{s_a\, dt}{q_a}\, dq \\ &= \frac{2}{\epsilon}\frac{w_i}{Q_i(w_i)}(1+\epsilon) s_a\, dt\end{aligned}$$

Here, the first inequality holds because $x/Q_i(x)$ is a non-decreasing function, and for all $q \in [0, q_a]$, we have $w_{a,i}(q) = w_{a,i}$ and $w_{o,i}(q) \leq w_{o,i}$ and hence $w_i(q) \geq w_i$.

Now to quantify the increase in the potential due to the optimal algorithm working: observe that for $q \in [0, q_o]$, the inner integral of Φ_i increases by at most

$$\int_{x=w_i(q)}^{w_i(q)+\frac{s_o\, dt}{q_o}} \frac{x}{Q_i(x)}\, dx = \frac{w_i(q)}{Q_i(w_i(q))}\frac{s_o\, dt}{q_o}$$

Again notice that we have used that fact that here dt is an infinitesimal period of time that in the limit is zero. Hence the total increase in Φ_i due to the optimal algorithm's processing is at most

$$\frac{2}{\epsilon} \int_{q=0}^{q_o} \frac{w_i(q)}{Q_i(w_i(q))}\frac{s_o\, dt}{q_o}\, dq \leq \frac{2}{\epsilon} \int_{q=0}^{q_o} \frac{w_i}{Q_i(w_i)}\frac{s_o\, dt}{q_o}\, dq = \frac{2}{\epsilon}\frac{w_i}{Q_i(w_i)} s_o\, dt.$$

Again here, the first inequality holds because $x/Q_i(x)$ is a non-decreasing function, and for all $q \in [0, q_o]$, we have $w_{a,i}(q) \leq w_{a,i}$ and $w_{o,i}(q) = w_{o,i}$ and hence $w_i(q) \leq w_i$.

Putting the two together, the overall increase in $\Phi_i(t)$ can be bounded by

$$\begin{aligned}\frac{d\Phi_i(t)}{dt} &\leq \frac{2}{\epsilon}\frac{w_{a,i}-w_{o,i}}{Q_i(w_{a,i}-w_{o,i})}\left[-(1+\epsilon)s_a+s_o\right]\\ &= \frac{2}{\epsilon}(w_{a,i}-w_{o,i})\frac{[-(1+\epsilon)Q_i(w_{a,i})+Q_i(w_{o,i})]}{Q_i(w_{a,i}-w_{o,i})}\\ &\leq -\frac{2}{\epsilon}\epsilon(w_{a,i}-w_{o,i}) = -2(w_{a,i}-w_{o,i})\end{aligned}$$

It is now easy to verify that by plugging this bound on $\frac{d\Phi_i(t)}{dt}$ into the running condition that one gets a valid inequality.

Case (3): $w_{a,i} = w_{o,i}$: In this case, let us just consider the increase due to OPT working. The inner integral in the potential function starts off from zero (since $w_{a,i} - w_{o,i} = 0$) and potentially (in the worst case) could increase to

$$\int_0^{\frac{s_o\,dt}{q_o}} \frac{x}{Q_i(x)}\,dx$$

(since $w_{o,i}$ drops by $s_o\,dt/q_o$ and $w_{a,i}$ cannot increase). However, since $x/Q_i(x)$ is a monotone non-decreasing function, this is at most

$$\int_0^{\frac{s_o\,dt}{q_o}} \frac{w_{o,i}}{Q_i(w_{o,i})}\,dx = \frac{s_o\,dt}{q_o}\frac{w_{o,i}}{Q_i(w_{o,i})}$$

Therefore, the total increase in the potential $\Phi_i(t)$ can be bounded by

$$\frac{2}{\epsilon}\int_{q=0}^{q_o}\frac{w_{o,i}}{Q_i(w_{o,i})}\frac{s_o\,dt}{q_o}\,dq = \frac{2}{\epsilon}s_o\,dt\frac{w_{o,i}}{Q_i(w_{o,i})} = \frac{2}{\epsilon}w_{o,i}\,dt$$

It is now easy to verify that by plugging this bound on $\frac{d\Phi_i(t)}{dt}$ into the running condition, and using the fact that $w_{a,i} = w_{o,i}$, one gets a valid inequality.

3 The Algorithm for Unweighted Flow

In this section, we give an immediate assignment based scheduling policy and the potential function that can be used to show (using basically the same line of reasoning as in the last section) that it is $O(1/\epsilon)$-competitive against a non-migratory adversary for the objective of *unweighted* flow plus energy, assuming the online algorithm has resource augmentation of $(1+\epsilon)$ in speed. Note that this result has a better competitiveness than the result for weighted flow from Section 2, but holds only for the unweighted case. Once again we can assume that the adversary is running the BCP algorithm on each processor. Just like the weighted case, the crux of our algorithm would be in designing a good assignment policy.

Our algorithm works as follows: Each processor maintains a queue of jobs that have currently been assigned to it. At some time instant t, for any processor i, let

$n^t_{a,i}(q)$ denote the number of jobs in processor i's queue that have a remaining processing time of at least q. Let $n^t_{a,i}$ denote the total number of unfinished jobs in the queue. Now our algorithm is the following:

When a new job arrives, the assignment policy assigns it to a processor which would cause the smallest increase in the "shadow potential"; i.e., a processor minimizing

$$\int_{q=0}^{p} \sum_{j=1}^{n^t_{a,i}(q)+1} \frac{j}{Q_i(j+1)}\,dq - \int_{q=0}^{p} \sum_{j=1}^{n^t_{a,i}(q)} \frac{j}{Q_i(j+1)}\,dq = \int_{q=0}^{p} \frac{(n^t_{a,i}(q)+1)}{Q_i(n^t_{a,i}(q)+2)}\,dq$$

The job selection on each processor is SRPT (Shortest Remaining Processing Time), and we set the power of processor i at time t to $n^t_{a,i}+2$ if $n^t_{a,i} \neq 0$ (and power zero otherwise). Once the job is assigned to a processor, it is never migrated.

Note that, even without accounting for resource augmentation, the online algorithm runs processor i at a speed of $Q_i(n^t_{a,i}+2)\cdot \mathbf{1}_{(n_{a,i}>0)}$, instead of $Q_i(n^t_{a,i}+1)\cdot \mathbf{1}_{(n_{a,i}>0)}$ as in [5]; since OPT uses BCP without any changes, it runs processor i at speed $Q_i(n^t_{o,i}+1)\cdot \mathbf{1}_{(n_{o,i}>0)}$.

We now describe our potential function Φ. For time t and processor i. Define $n^t_{o,i}$ as the number of unfinished jobs assigned to processor i by the optimal solution at time t, and $n^t_{o,i}(q)$ to be the number of these jobs with remaining processing time at least q. The global potential function is $\Phi(t) = \sum_i \Phi_i(t)$, where $\Phi_i(t)$ is the potential for processor i defined as:

$$\Phi_i(t) = \frac{4}{\epsilon} \int_{q=0}^{\infty} \sum_{j=1}^{(n^t_{a,i}(q)-n^t_{o,i}(q))_+} j/Q_i(j+1)\,dq \tag{4}$$

Recall that $(x)_+ = \max(x, 0)$, and $Q_i = P_i^{-1}$.

Acknowledgments. We thank Sangyeun Cho and Bruce Childers for helpful discussions about heterogeneous multicore processors, and also Srivatsan Narayanan for several useful discussions.

References

1. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. ACM Transactions on Algorithms 3(4) (2007)
2. Andrew, L.L., Wierman, A., Tang, A.: Optimal speed scaling under arbitrary power functions. SIGMETRICS Performance Evaluation Review 37(2), 39–41 (2009)
3. Bansal, N., Chan, H.L.: Weighted flow time does not admit o(1)-competitive algorithms. In: SODA, pp. 1238–1244 (2009)
4. Bansal, N., Chan, H.L., Lam, T.W., Lee, L.K.: Scheduling for speed bounded processors. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 409–420. Springer, Heidelberg (2008)

5. Bansal, N., Chan, H.L., Pruhs, K.: Speed scaling with an arbitrary power function. In: SODA, pp. 693–701 (2009)
6. Bansal, N., Pruhs, K., Stein, C.: Speed scaling for weighted flow time. SIAM Journal on Computing 39(4) (2009)
7. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Pruhs, K.: Online weighted flow time and deadline scheduling. J. Discrete Algorithms 4(3), 339–352 (2006)
8. Bower, F.A., Sorin, D.J., Cox, L.P.: The impact of dynamically heterogeneous multicore processors on thread scheduling. IEEE Micro 28(3), 17–25 (2008)
9. Chadha, J.S., Garg, N., Kumar, A., Muralidhara, V.N.: A competitive algorithm for minimizing weighted flow time on unrelatedmachines with speed augmentation. In: STOC, pp. 679–684 (2009)
10. Chan, H.L., Edmonds, J., Lam, T.W., Lee, L.K., Marchetti-Spaccamela, A., Pruhs, K.: Nonclairvoyant speed scaling for flow and energy. In: STACS, pp. 255–264 (2009)
11. Greiner, G., Nonner, T., Souza, A.: The bell is ringing in speed-scaled multiprocessor scheduling. In: SPAA 2009: Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures, pp. 11–18. ACM, New York (2009)
12. Kumar, R., Tullsen, D.M., Jouppi, N.P.: Core architecture optimization for heterogeneous chip multiprocessors. In: International conference on parallel architectures and compilation techniques, pp. 23–32. ACM, New York (2006)
13. Kumar, R., Tullsen, D.M., Ranganathan, P., Jouppi, N.P., Farkas, K.I.: Single-isa heterogeneous multi-core architectures for multithreaded workload performance. SIGARCH Computer Architecture News 32(2), 64 (2004)
14. Lam, T.W., Lee, L.K., To, I.K.K., Wong, P.W.H.: Competitive non-migratory scheduling for flow time and energy. In: SPAA, pp. 256–264 (2008)
15. Lam, T.W., Lee, L.K., To, I.K.K., Wong, P.W.H.: Speed scaling functions for flow time scheduling based on active job count. In: European Symposium on Algorithms, pp. 647–659 (2008)
16. Leonardi, S., Raz, D.: Approximating total flow time on parallel machines. Journal of Computer and Systems Sciences 73(6), 875–891 (2007)
17. Merritt, R.: CPU designers debate multi-core future. EE Times (February 2008)
18. Morad, T.Y., Weiser, U.C., Kolodny, A., Valero, M., Ayguade, E.: Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors. IEEE Computer Architecture Letters 5(1), 4 (2006)
19. Pruhs, K.: Competitive online scheduling for server systems. SIGMETRICS Performance Evaluation Review 34(4), 52–58 (2007)
20. Pruhs, K., Sgall, J., Torng, E.: Online scheduling. In: Handbook on Scheduling, CRC Press, Boca Raton (2004)
21. Pruhs, K., Uthaisombut, P., Woeginger, G.J.: Getting the best response for your erg. ACM Transactions on Algorithms 4(3) (2008)

Better Scalable Algorithms for Broadcast Scheduling

Nikhil Bansal[1], Ravishankar Krishnaswamy[2,⋆], and Viswanath Nagarajan[1]

[1] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
[2] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract. In the classic *broadcast scheduling problem*, there are n pages stored at a server, and requests for these pages arrive over time. Whenever a page is broadcast, it satisfies all outstanding requests for that page. The objective is to minimize the average *flowtime* of the requests. In this paper, for any $\epsilon > 0$, we give a $(1+\epsilon)$-speed $O(1/\epsilon^3)$-competitive online algorithm for the broadcast scheduling problem, even when page sizes are not identical. This improves over the recent breakthrough result of Im and Moseley [18], where they obtained a $(1+\epsilon)$-speed $O(1/\epsilon^{11})$-competitive algorithm for the setting when all pages have the same size.

This is the first scalable algorithm for broadcast scheduling with varying size pages, and resolves the main open question from [18]. Furthermore, our algorithm and analysis are considerably simpler than [18], and also extend to the general setting of *dependent requests*.

1 Introduction

We consider the classic problem of scheduling in a broadcast setting to minimize the average response time. In the broadcast scheduling problem, there are n pages, and requests for these pages arrive over time. There is a single server that can broadcast pages. Whenever a page is transmitted, it satisfies all outstanding requests for that page. In the most basic version of the problem, we assume that time is slotted and that each page can be broadcast in a single time slot. Any request r is specified by its arrival time $a(r)$ and the page $p(r)$ that it requests; we let $[m]$ denote the set of all requests. A broadcast schedule is an assignment of pages to time slots. The flow-time (or response time) of request r under a broadcast schedule equals $b(r) - a(r)$ where $b(r) \geq a(r)+1$ is the earliest time slot after $a(r)$ when page $p(r)$ is broadcast. The objective is to minimize the average flow-time, i.e. $\frac{1}{m} \cdot \sum_{r \in [m]} (b(r) - a(r))$.

More general versions of the problem have also been studied. One generalization is to assume that pages have different sizes. A complicating issue in this case is that a request for a page may arrive in the midst of a transmission of this page. There are two natural models studied here, depending on whether the client can cache content or not. In the caching version, a request is considered satisfied as soon as it sees one complete transmission of a page (so it could first receive the latter half of the page and then receive the first half). Without a cache, a request can only be satisfied when it starts receiving the page from the beginning and when the page is completely transmitted. The latter

⋆ Supported in part by NSF awards CCF-0448095 and CCF-0729022, and an Alfred P. Sloan Fellowship. Work done while author was visiting IBM T. J. Watson Research Center.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 324–335, 2010.

version is natural, for example, with movie transmissions, while the former is applicable for file transmissions. When pages have arbitrary sizes, it is also standard to consider preemptive schedules (i.e. transmission of a page need not occur at consecutive time-slots). This is because no reasonable guarantee can exist if preemption is disallowed.

Another generalization is the case of so called dependent requests. Here a request consists of a subset of pages, and this request is considered completed only when all the pages for this request have been broadcast.

1.1 Previous Work

The broadcast scheduling setting has studied extensively in the last few years, both in the offline and online setting. Most of the work has been done on the most basic setting with unit page sizes and no dependencies. In addition to minimizing the average response time, various other metrics such as throughput maximization [6,21,15], maximum response time [4,5,8,9], delay-factor [9] etc. have also been studied quite extensively. We describe here the work related to minimizing the average response time. We first consider the offline case. The first guarantee of any kind was a 3-speed, 3-approximation due to Kalyanasundaram, Pruhs and Veluthapillai [20]. After a sequence of works [16,17,1], an $O(\log^2 n/\log\log n)$-approximation based on iterated rounding techniques was obtained by Bansal, Coppersmith and Sviridenko [2]. This is currently the best approximation known for the problem. It is also known that the problem is NP-Hard [14,5]. While no APX-hardness result is known, it is known that the natural LP formulation (which is the basis of all known results for this problem), has a (rather small) integrality gap of $28/27 = 1.037$ [1].

In the online case, which is perhaps more interesting for practical applications of the problem, very strong lower bounds are known. In particular, any deterministic algorithm must be $\Omega(n)$ competitive and any randomized algorithm must be $\Omega(\sqrt{n})$ competitive [20,1]. Thus, it is most natural to consider the problem in the resource augmentation setting, where the online algorithm is provided a slightly faster server than the optimum offline algorithm. The first positive result was due to Edmonds and Pruhs [11] who gave an algorithm B-Equi and showed that it is $(4+\epsilon)$-speed, $O(1/\epsilon)$-competitive. The algorithm B-Equi produced a schedule where several pages may be transmitted fractionally in a single time slot. Edmonds and Pruhs [11] also showed how to convert B-Equi into a valid schedule (i.e. only one page is transmitted in each time slot) using another $(1+\epsilon)$-speedup and losing a factor of $1/\epsilon$ in the competitive ratio, which gave a $(4+\epsilon)$-speed, $O(1/\epsilon^2)$-competitive algorithm.

This result is based on a very interesting idea. The authors make a connection with another scheduling problem on multiprocessors known as non-clairvoyant scheduling with sublinear-nondecreasing speedup curves. This problem is very interesting in its own right with several applications and was introduced in a previous paper by Edmonds [10]. In that paper, Edmonds gave a $(2+\epsilon)$-speed, $O(1/\epsilon)$-competitive algorithm called Equi for the non-clairvoyant scheduling problem. Edmonds and Pruhs showed that the broadcast scheduling problem can be reduced to non-clairvoyant scheduling problem while losing a factor of 2 in the speed up required [11]. Given the $(2+\epsilon)$-speed, $O(1/\epsilon)$-competitive algorithm Equi, this yields the $(4+\epsilon)$-speed, $O(1/\epsilon)$-algorithm B-Equi for broadcast (where pages are transmitted fractionally in each time-slot).

Recently, Edmonds and Pruhs [13] gave a very elegant algorithm called $\mathsf{LAPS}(\beta)$ for the non-clairvoyant scheduling problem. They showed that for any $\epsilon > 0$, the algorithm $\mathsf{LAPS}(\epsilon/2)$ is $(1 + \frac{\epsilon}{2})$-speed $O(1/\epsilon^2)$ competitive. Using the Edmonds-Pruhs reduction from broadcast to non-clairvoyant scheduling mentioned above [11], this implies an $(2 + \epsilon)$-speed, $O(1/\epsilon^2)$-competitive 'fractional' broadcast schedule. Losing another factor of $1/\epsilon$, this can be converted to a valid broadcast schedule that is $(2 + \epsilon)$-speed, and $O(1/\epsilon^3)$-competitive. These results [11,13] also hold when page sizes are non-unit but preemption is allowed.

Another natural online algorithm that has been studied is Longest Wait First (LWF). This is a natural greedy algorithm that at any time broadcast the page for which the total waiting time of outstanding requests is the highest. Edmonds and Pruhs [12] showed that LWF is 6-speed, $O(1)$-competitive. They also showed that no $n^{o(1)}$ guarantee is possible unless the speedup is at least $(1 + \sqrt{5})/2 \approx 1.61$. In particular, this rules out the possibility of LWF being a $(1 + \epsilon)$-speed, $O(1)$-competitive algorithm. Recently, the results for LWF has been improved by [7]. They show that LWF if 2.74-speed, $O(1)$-competitive. They also improve the lower bound on speed up required to $2 - \epsilon$.

Until recently, a major open question in the area had been whether there are fully scalable algorithms. Intuitively, fully scalable algorithms are important from a practical point of view, since one would expect them to perform close to optimum in practice. See [19,22] for a formal discussion of this issue. Recently, in a breakthrough result, Im and Moseley [18] obtained the first scalable algorithms for broadcast scheduling. In particular, they design an algorithm call $LA-W$, that is $(1+\epsilon)$-speed, $O(1/\epsilon^{11})$-competitive. This algorithm is similar to LWF, but it favors pages that have recent requests. The analysis of $LA - W$ is based on a rather complicated charging scheme. Additionally, the algorithm in [18] only works for unit-size pages, and the authors leave open the question for varying-size pages.

The case of dependent requests has been studied by [23]. They show that a generalization of the B-Equi algorithm, called B-EquiSet is $(4+\epsilon)$-speed, $O(1/\epsilon^3)$-competitive, even in the setting where pages have arbitrary lengths (with preemptions).

1.2 Our Results

In this paper we give fully scalable algorithms for broadcast scheduling with improved guarantees. Our algorithm and analysis are much simpler than that of [18], and they also extend to the general setting with non-uniform page sizes and dependent requests. In particular we prove the following results:

Theorem 1. *Consider the broadcast scheduling setting where pages have arbitrary sizes and requests are dependent. Moreover, no cache is available. Then, if preemption is allowed, for every* $0 < \epsilon \leq 1/4$*, there is a* $(1 + \epsilon)$*-speed,* $O\left(\frac{1}{\epsilon^3}\right)$*-competitive deterministic online algorithm.*

Again, when we say there is no cache available, it means that a request for a page is satisfied only at the earliest time when the contents of the page are received in order. In particular, if a request for a page p occurs in the middle of some broadcast of p, it has to wait for the current broadcast of p to complete before receiving the next broadcast

of the page in the correct order. In the model with dependent requests, each request is associated with a *set of pages*, and is satisfied only when each of the pages in its set has been transmitted in correct order.

As mentioned earlier, in order to obtain any reasonable guarantees under arbitrary page-sizes, one needs to consider the preemptive version, i.e. transmissions of a page need not occur continuously (for an example as to why, refer to the full version of the paper [3]).

Remark: Although we state the above result only for the (more restrictive) version where there is no cache available, the approximation guarantee in Theorem 1 holds relative to an optimal schedule for the (less restrictive) version where cache is available. Thus Theorem 1 implies scalable online algorithms for both versions, with and without caching. For the setting where cache is available, the problem can even be reduced to dependent requests with unit sized pages, since we can replace a page p of length ℓ_p by ℓ_p unit size pages, and modify any original request for p to now request the corresponding ℓ_p unit size pages.

In the full version [3], we also prove the following theorem, which shows a (randomized) algorithm with better competitive ratio for the special case when all pages have identical sizes.

Theorem 2. *If all pages are of unit size, then for every* $0 < \epsilon \leq 1/4$*, there is a* $(1+\epsilon)$*-speed,* $O\left(\frac{1}{\epsilon^2}\right)$*-competitive randomized online algorithm for broadcast scheduling.*

Our algorithms and its analysis are inspired by the algorithm LAPS for non-clairvoyant scheduling [13]. Our main idea is to bypass the reduction (from broadcast scheduling to non-clairvoyant scheduling) [11] that loses a factor of 2 in the speedup and directly adapt those ideas to broadcast scheduling. The algorithm and its analysis are in fact very simple. Our approach is the following: We first consider the *fractional* version of the problem (i.e. pages can be fractionally transmitted in each time-slot and a request for a page of size ℓ is satisfied as soon as there have been ℓ units of the page transmitted after the arrival of the request) and show that a variant of LAPS (adapted to the broadcast setting) is $(1+\epsilon)$-speed, $O(1/\epsilon^2)$-competitive. Note that this guarantee matches that for LAPS. Then we show how to round this fractional schedule in an online manner to obtain an integral schedule (i.e. only one page transmitted in each time-slot, and a request is satisfied only when it receives the contents of the page *in order*). This idea of reducing broadcast scheduling to a fractional version, and solving the fractional version was also used implicitly in the algorithms of Edmonds and Pruhs [11,12], but one main difference in our work is that we consider a different fractional relaxation, and this enables us to obtain a fully scalable algorithm.

Given the results above, a natural question is whether the loss of factor 2 speed up in previous approaches [11,12] can be avoided in the reduction from broadcast scheduling to the non-clairvoyant scheduling problem. It turns out that this is indeed possible. We give a reduction from fractional broadcast scheduling to non-clairvoyant scheduling that does not incur any loss in speed up or in the competitive ratio (i.e. it is a $(1,1)$ transformation). Again, the main idea to achieve this lies in the appropriate definition of the fractional broadcast problem, and the online rounding algorithms required to reduce the broadcast problem to its fractional relaxation. Note that this reduction combined with LAPS [13] would also imply our results. However, in this paper we have chosen to

present our results directly without going via the non-clairvoyant reduction, since the proofs seem much simpler and cleaner with this approach. We note that this reduction could be useful in other contexts, and present it in the full version.

2 Broadcast Scheduling: The General Model

We now formally define the problem we are studying. There are n pages with each page p having an integer size l_p; page p consists of l_p distinct units/positions that are numbered 1 to l_p (for instance, one could think of these units as the individual bytes of the page). Requests for *subsets* of these pages arrive over time. There is a single server that can broadcast pages: in each time-slot the server broadcasts some position/unit of some page. Any request r is specified by its arrival time $a(r)$ and the set of pages $\mathcal{P}(r) \subseteq [n]$ that it requests; we let $[m]$ denote the set of all requests. A broadcast schedule is an assignment of page-positions (i.e. tuple $\langle p, i\rangle$ where $p \in [n]$ and $i \in \{1, \cdots, l_p\}$) to time slots. For any request r, page $p \in \mathcal{P}(r)$ is said to be *completed* if the server has broadcast all the l_p positions of page p in the order 1 through l_p, after the arrival time time $a(r)$ (though these may not necessarily occur in consecutive time-slots). The flow-time of request r under a broadcast schedule equals $b(r) - a(r)$ where $b(r) \geq a(r) + 1$ is the earliest time slot after $a(r)$ when *all* the pages requested in $\mathcal{P}(r)$ have been completed. The objective is to minimize the average flow-time, i.e. $\frac{1}{m} \cdot \sum_{r \in [m]} (b(r) - a(r))$. We assume that the pages all have size at least 1, and therefore the optimal value is also at least one.

3 Fractional Broadcast Scheduling

As mentioned earlier, our algorithm is based on first solving the 'fractional' version of the problem, and then rounding this fractional schedule into a valid 'integral' schedule. Recall that with non-uniform page sizes, an integral schedule is one where only one page is transmitted in each time slot; however since pages have arbitrary sizes, complete transmission of a page may occupy non-contiguous time-slots.

In the fractional broadcast problem, the algorithm can transmit pages in a continuous manner. Here, at any (continuous) time instant t, the algorithm is allowed to broadcast each page $p \in [n]$ at rate $x_p(t)$, such that $\sum_{p=1}^{n} x_p(t) \leq 1$ for all t. In the setting with resource augmentation, we allow $\sum_p x_p(t) \leq 1 + \epsilon$ for all t (i.e the online algorithm gets a slightly higher transmission rate). For any request $r \in [m]$ and page $p \in \mathcal{P}(r)$, define

$$b(r,p) := \inf \left\{ s \ : \ \int_{a(r)}^{s} x_p(t)dt \geq l_p \right\},$$

i.e. the earliest time after the release of request r when l_p *units* of page p have been broadcast. The *fractional completion time* of any request $r \in [m]$ is then:

$$b(r) := \max_{p \in \mathcal{P}(r)} b(r,p),$$

i.e. the time after the arrival of request r when all pages requested by r have been completely broadcast. Finally the flow-time of request r equals $b(r) - a(r)$. Notice that

in fractional broadcast (unlike the original broadcast problem), we don't have the notion of l_p distinct positions of each page p, and also don't have the requirement of receiving a page in order; we only require the schedule to broadcast a total of l_p indistinguishable units for page p.

3.1 The Fractional Algorithm

We now describe our algorithm. At any continuous time t, let $N(t)$ denote the set of *active requests*, i.e. those which have not yet been fractionally completed. Let $N'(t)$ denote the $\epsilon|N(t)|$ "most-recent" requests among $N(t)$, i.e. those with the latest arrival times. For each request $r \in N'(t)$, let $\mathsf{Unfin}(r,t)$ denote an arbitrary page $p \in \mathcal{P}(r)$ that has not been fractionally broadcast to an extent l_p since the arrival time $a(r)$. The algorithm then time shares among the pages $\{\mathsf{Unfin}(r,t) \mid r \in N'(t)\}$, i.e.

$$x_p(t) := (1+4\epsilon) \cdot \frac{|\{r \in N'(t) : \mathsf{Unfin}(r,t) = p\}|}{|N'(t)|}, \quad \forall\, p \in [n].$$

Clearly, $\sum_{p=1}^{n} x_p(t) \leq 1 + 4\epsilon$ at all times t. For the sake of analysis, also define:

$$y_{r,p}(t) := \begin{cases} \frac{1+4\epsilon}{|N'(t)|} & \text{if } r \in N'(t), \text{ and } p = \mathsf{Unfin}(r,t) \\ 0 & \text{otherwise} \end{cases}, \qquad \forall\, t \geq 0$$

In particular, $y_{r,p}(t)$ is the share of request of r for page p, if we distribute the $1+4\epsilon$ processing equally among requests in $N'(t)$. Notice that the rate $x_p(t)$ of page p transmitted at time t could be much larger than $y_{r,p}(t)$ even for requests with $\mathsf{Unfin}(r,t) = p$, because $\mathsf{Unfin}(r',t)$ could also be page p for other requests r'.

3.2 Analysis of Fractional Broadcast

Our analysis is based on a potential function argument inspired by that for LAPS [13]. We first describe the potential function, and then use it to bound the competitive ratio.

Let Opt denote an optimal (offline) *fractional* broadcast schedule for the given instance, and let On denote the fractional online schedule produced by the above algorithm. For any request $r \in [m]$, let $b^*(r)$ denote the completion time of r in Opt, and let $b(r)$ denote its completion time in On. For any $r \in [m]$, page $p \in \mathcal{P}(r)$, and times $t_1 < t_2$, define $\mathsf{On}(r,p,t_1,t_2) := \int_{t_1}^{t_2} y_{r,p}(t)dt$, i.e. the fractional time that the online algorithm has devoted towards *page p on behalf of request r* in the interval $[t_1,t_2]$. Observe that since any request r is inactive after time $b(r)$, we have $y_{r,p}(t) = 0$ for all $t > b(r)$ and $p \in \mathcal{P}(r)$. Thus $\mathsf{On}(r,p,t,\infty) = \mathsf{On}(r,p,t,b(r))$ for all $r \in [m]$, $p \in \mathcal{P}(r)$, and $t \geq 0$.

At any (continuous) time t and for any page $p \in [n]$, let $x^*_p(t)$ denote the rate at which Opt broadcasts p. We have $\sum_p x^*_p(t) \leq 1$ since the offline optimal is 1-speed. For page $p \in [n]$ and times $t_1 < t_2$, let $\mathsf{Opt}(p,t_1,t_2) := \int_{t_1}^{t_2} x^*_p(t)dt$ denote the amount of page p transmitted by Opt in the interval $[t_1,t_2]$. At any continuous time t, let $N(t)$ and $N^*(t)$ denote the set of requests that are not completed in On and Opt respectively.

We now define the contribution of any request $r \in [m]$ and page $p \in \mathcal{P}(r)$ to the potential as follows.

$$z_{r,p}(t) = \frac{\mathsf{On}(r,p,t,\infty) \cdot \mathsf{Opt}(p,a(r),t)}{l_p}$$

The total contribution of request r is then $z_r(t) = \sum_{p \in \mathcal{P}(r)} z_{r,p}(t)$. Note that $z_r(t) \geq 0$ for any r and t. Finally, the overall potential function is defined as

$$\Phi(t) := \frac{1}{\epsilon} \cdot \sum_{r \in N(t)} \mathsf{rank}(r) \cdot z_r(t),$$

where rank is the function which orders active requests based on arrival times (with the highest rank of $|N(t)|$ going to the request which arrived most recently and a rank of 1 to the oldest active request).

We will now show that the following inequality holds over all sufficiently small intervals $[t, t+dt)$ such that no requests arrive or complete in On during this interval. Time instants where requests arrive or complete in On will be handled separately.

$$\Delta\mathsf{On}(t) + \Delta\Phi(t) \leq \frac{2}{\epsilon^2}\Delta\mathsf{Opt}(t). \tag{3.1}$$

Since we ensure that $\Phi(0) = \Phi(\infty) = 0$, it is immediate to see that the total cost of the online algorithm is competitive with the optimal offline cost, up to a factor of $\frac{2}{\epsilon^2}$.

Request Arrival: We show that $\Delta\Phi = 0$ (clearly this suffices, since we can assume that arrivals happen instantaneously and hence $\Delta\mathsf{On} = \Delta\mathsf{Opt} = 0$). When a request r arrives at time t, we have $z_r(t) = 0$ as r is entirely unsatisfied by Opt. Thus, Φ does not change due to r. Moreover, as the requests are ranked in the increasing order of their arrival, the ranks of other requests are unaffected and hence $\Delta\Phi = 0$.

Request Completes under Online and leaves the set $N(t)$: When a request r leaves $N(t)$, by definition its $z_r(t)$ reaches 0. Moreover, the rank of any other request $r' \in N(t)$ can only decrease. Since $z_{r'}(t) \geq 0$ for any r', the contribution due to these requests to the potential can only decrease. Thus $\Delta\Phi \leq 0$. And again, at that instant, $\Delta\mathsf{On} = \Delta\mathsf{Opt} = 0$, and hence equation (3.1) holds.

Now consider any sufficiently small interval $(t, t+dt)$ when neither of the above two events happen. There are two causes for change in potential:

Offline broadcast in $(t, t+dt)$: We will show that $\Delta\Phi(t) \leq \frac{1}{\epsilon}|N(t)|dt$. To see this, consider any page p. The amount of page p transmitted by Opt in this interval is $x^*_p(t)dt$. This broadcast of $x^*_p(t)dt$ amount of page p causes the quantity $z_{r,p}(t)$ to increase for all those requests r that are alive and have $p \in \mathcal{P}(r)$ unfinished in On at time t. Recall the definition of 'completion time' $b(r,p)$ for page p of request r. Define,

$$C(t,p) := \{r \in [m] \mid p \in \mathcal{P}(r),\ a(r) \leq t < b(r,p)\}$$

Now, since the rank of any alive request is at most $|N(t)|$, we get that the total increase in Φ over the interval $[t, t+dt)$ due to Opt's broadcast is at most:

$$\Delta\Phi \leq \frac{1}{\epsilon}|N(t)| \cdot \sum_p \sum_{r \in C(t,p)} \frac{\mathsf{On}(r,t,\infty)x^*_p(t)dt}{l_p}. \tag{3.2}$$

We show that $\sum_{r\in C(t,p)} \mathsf{On}(r,t,\infty) \le l_p$ for any page p. Let $r' = \arg\max\{b(r,p) \mid r \in C(t,p)\}$ be the request in $C(t,p)$ for which page p is completed last by On. Since page p for r' is *not* completed until $b(r',p)$ and $a(r') \le t$, it must be that On broadcasts at most l_p units of page p during $[t, b(r',p)]$; otherwise $b(r',p)$ would be smaller. Hence $\sum_{r\in C(t,p)} \mathsf{On}(r,t,b(r',p)) \le l_p$. Observe that for all $r \in C(p,t)$ and $t \ge b(r',p)$, we have $y_{r,p}(t) = 0$ since $b(r,p) \le b(r',p)$. Thus $\sum_{r\in C(t,p)} \mathsf{On}(r,t,\infty) \le l_p$. Now plugging this into equation (3.2), we have that

$$\Delta\Phi \le \frac{1}{\epsilon}|N(t)| \cdot \sum_p x^*_p(t)dt \le \frac{1}{\epsilon}|N(t)| \cdot dt \tag{3.3}$$

Recall that $\sum_p x^*_p(t) \le 1$ since Opt is 1-speed.

Online broadcast in $(t, t+dt)$*:* Recall that On broadcasts page p at rate $x_p(t)$, and $y_{r,p}(t)$ is the rate at which On works on page p for request r. Consider any fixed request $r \in N'(t) \setminus N^*(t)$, i.e. on which On works but is completed by Opt. Observe that for every $p \in \mathcal{P}(r)$, $\mathsf{Opt}(r,p,t) \ge l_p$ since Opt has completed request r. Thus $z_r(t) \ge \sum_{p\in\mathcal{P}(r)} \mathsf{On}(r,p,t,\infty)$. Note also that $\sum_{p\in\mathcal{P}(r)} y_{r,p}(t) = (1+4\epsilon)/|N'(t)|$. Thus,

$$\frac{d}{dt}z_r(t) \le -\sum_{p\in\mathcal{P}(r)} y_{r,p}(t) = -\frac{1+4\epsilon}{|N'(t)|}, \quad \text{for all } r \in N'(t) \setminus N^*(t).$$

Furthermore, since each request that On works on in $[t, t+dt)$ has rank at least $(1-\epsilon)\cdot|N(t)|$, the potential Φ increases at rate,

$$\frac{d}{dt}\Phi(t) \le -\frac{1}{\epsilon}(1-\epsilon)N(t) \cdot \frac{(1+4\epsilon)}{|N'(t)|}\left(|N'(t)| - |N^*(t)|\right).$$

Since $(1-\epsilon)(1+4\epsilon) \ge (1+2\epsilon)$ for $\epsilon \le 1/4$, we get

$$\frac{d}{dt}\Phi(t) \le -\left(\frac{1}{\epsilon}+2\right)|N(t)| + \frac{1}{\epsilon^2}(1+4\epsilon)|N^*(t)| \le -\left(\frac{1}{\epsilon}+1\right)\cdot|N(t)| + \frac{2}{\epsilon^2}\cdot|N^*(t)|. \tag{3.4}$$

Observe that $\frac{d}{dt}\mathsf{On}(t) = |N(t)|$ and $\frac{d}{dt}\mathsf{Opt}(t) = |N^*(t)|$. Using (3.3) and (3.4),

$$\begin{aligned}\frac{d}{dt}\mathsf{On}(t) + \frac{d}{dt}\Phi(t) &\le |N(t)| + \frac{1}{\epsilon}|N(t)| - \left(\frac{1}{\epsilon}+1\right)|N(t)| + \frac{2}{\epsilon^2}|N^*(t)| \\ &\le \frac{2}{\epsilon^2}|N^*(t)| \;=\; \frac{2}{\epsilon^2}\cdot\frac{d}{dt}\mathsf{Opt}(t),\end{aligned}$$

which proves Equation (3.1). Thus we obtain:

Theorem 3. *For any* $0 < \epsilon \le \frac{1}{4}$*, there is a* $(1+4\epsilon)$*-speed* $O\left(\frac{1}{\epsilon^2}\right)$ *competitive deterministic online algorithm for fractional broadcast scheduling with dependencies and non-uniform sizes.*

4 Deterministic Online Rounding of Fractional Broadcast

In this section, we focus on getting an integral broadcast schedule from the fractional schedule in an online deterministic fashion. Formally, given any 1-speed fractional

broadcast schedule On, we will obtain a $(1+\epsilon)$-speed integral broadcast schedule Rnd (which gets to transmit an *additional unit of page* every $\lceil \frac{1}{\epsilon} \rceil$ time-steps) such that

$$\sum_r (b_I(r) - a(r)) \leq O\left(\frac{1}{\epsilon}\right) \cdot \sum_r (b(r) - a(r)) \tag{4.5}$$

where $b_I(r)$ (resp. $b(r)$) is the completion time of request r in the integral (resp. fractional) schedule. An important issue in converting the fractional schedule to an integral one is that a valid broadcast of any page p now requires the l_p positions of page p to be transmitted in the correct order. While this is relatively easy to guarantee if one is willing to lose a factor of 2 in the speed up, (see for example the rounding step in [11,23]), the algorithm here is much more subtle.

For any request $r \in [m]$ and page $p \in \mathcal{P}(r)$, *job* $\langle r, p\rangle$ denotes the page p request due to r. The arrival time of job $\langle r, p\rangle$ is the arrival time $a(r)$ of the corresponding request. We say that a job $\langle r, p\rangle$ is completed if the schedule contains a valid broadcast of page p starting after time $a(r)$. The completion time of job $\langle r, p\rangle$ in schedule Rnd (resp. On) is denoted $b_I(r, p)$ (resp. $b(r, p)$). The rounding algorithm maintains a queue of tuples (denoting transmissions of pages) of the form $\tau = \langle p, w, s, i\rangle$ where $p \in [n]$ is a page, $w \in \mathbb{R}_+$ is the *width*, $s \in \mathbb{Z}_+$ is the *start-time*, and $i \in \{1, \cdots, l_p\}$ is the *index* of the next position of page p to transmit. At each time-slot, the deterministic schedule broadcasts the current position of the tuple having *least width*. Intuitively, if the fractional solution completed a page very quickly, we would want the corresponding transmission of that page to occur quickly in the integral solution as well. Also, for each page we need to track the time s when the current transmission began for this page, and the fraction of this page transmitted since time s to ensure that the requests receive the pages in correct order. Formally, our algorithm is described in Algorithm 1.

In order to bound the flowtime in schedule Rnd, we prove the following:

$$b_I(r, p) - b(r, p) \leq \frac{6}{\epsilon} \cdot \big(b(r, p) - a(r)\big) + \frac{10}{\epsilon}, \quad \text{for all jobs } \langle r, p\rangle. \tag{4.6}$$

Notice that since this applies for all pages corresponding to any request, this implies the stated goal in equation (4.5), since $b_I(r, p) - a(r) = b_I(r, p) - b(r, p) + b(r, p) - a(r)$. To this end, consider any fixed job $\langle r, p\rangle$, and let $t = b(r, p)$. If at this time t, job $\langle r, p\rangle$ is *marked* then clearly $b_I(r, p) \leq t = b(r, p)$ and Equation (4.6) holds. So assume that $\langle r, p\rangle$ is unmarked. In this case (from the description of the algorithm) it must be that $\mathcal{Q}$ contains a tuple $\tau = \langle p, w, s, i\rangle$ where width $w \leq b(r, p) - a(r)$. Then let us define,

$$t_A \coloneqq \max\{z \leq t \mid \text{ at time } z, \text{either some request of width} > w \text{ is dequeued or } \mathcal{Q}{=}\emptyset\}$$

$$t_B \coloneqq \min\{z \geq t \mid \text{ at time } z, \text{either some request of width} > w \text{ is dequeued or } \mathcal{Q}{=}\emptyset\}$$

Hence schedule Rnd always broadcasts some tuple of width at most w during interval $\mathcal{T} := (t_A, t_B)$, and there are no tuples of width smaller than w at times t_A and t_B. Clearly $b_I(r, p) \leq t_B$ and $b(r, p) = t \geq t_A$; so it suffices to upper bound $t_B - t_A$ by the right hand side in (4.6).

Fix a page $q \in [n]$, and let Π_q denote the set of all tuples of page q that are broadcast (in even one time-slot) during $\mathcal{T}$. Let $N_q = |\Pi_q|$. We now prove some claims about Π_q.

Algorithm 1. GenRounding(t)

1: **initialize** all jobs as unmarked when they arrive.
2: **simulate** fractional online algorithm to obtain schedule On.
3: **for** any *unmarked* job $\langle r, p\rangle$ that completes under On at time t, i.e. $b(r,p) = t$, **do**
4: **if** there is a tuple $\tau = \langle p, w, s, i\rangle \in \mathcal{Q}$ of page p with $s \geq a(r)$ **then**
5: **update** the width of tuple τ to $\min(w, b(r,p) - a(r))$.
6: **else**
7: **insert** new tuple $\langle p, b(r,p) - a(r), \infty, 1\rangle$ into $\mathcal{Q}$.
8: **end if**
9: **end for**
10: **dequeue** the tuple $\tau = \langle p, w, s, i\rangle$ that has least width amongst all elements in $\mathcal{Q}$.
11: **broadcast** position i of page p in this time-slot.
12: **if** broadcast of p corresponding to τ is just beginning (i.e. $i = 1$) **then**
13: **set** $s = t$, i.e. equal to the current time slot .
14: **end if**
15: **if** broadcast of p corresponding to τ is complete (i.e. $i = l_p$) **then**
16: **mark** all jobs $\langle r', p\rangle$ of page p having $a(r') \leq s$.
17: **else**
18: **enqueue** the modified tuple $\langle p, w, s, i+1\rangle$ into $\mathcal{Q}$.
19: **end if**
20: **repeat** steps (10)-(19) if t is an integer multiple of $\lceil \frac{1}{\epsilon} \rceil$.

Lemma 1. *For each $\tau \in \Pi_q$, the start-time $s(\tau) \geq t_A - w$.*

Proof. Since τ is broadcast at some time-slot during $\mathcal{T}$, its width must be at most w at that time. Let $\langle r', q\rangle$ denote the job that caused τ's width to be at most w. Then it must be that $a(r') \leq s(\tau)$ and $b(r', q) \leq a(r') + w \leq s(\tau) + w$. Observe that at time t_A, queue $\mathcal{Q}$ contains no tuple of width at most w. Thus $b(r', q) \geq t_A$, i.e. $s(\tau) \geq t_A - w$, which proves the lemma. ■

Based on this lemma, we index tuples in Π_q as $\{\tau_j \mid 1 \leq j \leq N_q\}$ in increasing order of the start-times, i.e. $t_A - w \leq s(\tau_1) \leq s(\tau_2) \leq \cdots s(\tau_{N_q}) \leq t_B$. In the following, for page q and times $t_1 < t_2$, $\mathsf{On}(q, t_1, t_2)$ denotes the amount of page q transmitted by fractional schedule On during interval (t_1, t_2).

Lemma 2. *For any $1 \leq j \leq N_q - 1$, we have $\mathsf{On}(q, s(\tau_j), s(\tau_{j+1})) \geq l_q$.*

Proof. Consider the time t' when tuple τ_{j+1} is first inserted into $\mathcal{Q}$. Since τ_j must have entered $\mathcal{Q}$ before τ_{j+1}, it must be that $s(\tau_j) < t' \leq s(\tau_{j+1})$; otherwise τ_{j+1} would not be inserted as a new tuple. Suppose τ_{j+1} is inserted due to the completion of job $\langle r', q\rangle$ in On. Then it must also be that $a(r') > s(\tau_j)$; otherwise job $\langle r', q\rangle$ would just have updated the width of τ_j and not inserted a new tuple in Step 7. Clearly $b(r', q) = t'$, and hence $\mathsf{On}(q, s(\tau_j), s(\tau_{j+1})) \geq \mathsf{On}(q, a(r'), b(r', q)) \geq l_q$. ■

Lemma 3. $\mathsf{On}(q, t_A - w, t_C) \geq l_q$, *where* $t_C = \max\{s(\tau_1), t_A + w\}$.

Proof. Let $\langle r', q\rangle$ denote the *first* job that caused τ_1's width to be at most w (recall from lemma 1, there must be such a job). Again, it must be that $b(r', q) \geq t_A$ and so $a(r') \geq t_A - w$. We consider two cases:

1. $s(\tau_1) \leq t_A$. In this case, we have $a(r') \leq s(\tau_1) \leq t_A$ and so $b(r', q) \leq a(r') + w \leq t_A + w$. Thus $\mathsf{On}(q, t_A - w, t_A + w) \geq \mathsf{On}(q, a(r'), b(r', q)) \geq l_q$.
2. $s(\tau_1) > t_A$. Since start-time of tuple τ_1 lies in $\mathcal{T}$, for job $\langle r', q\rangle$ we have $b(r', q) \leq s(\tau_1)$. Therefore, $\mathsf{On}(q, t_A - w, s(\tau_1)) \geq \mathsf{On}(q, a(r'), b(r', q)) \geq l_q$.

Since $t_C = \max\{s(\tau_1), t_A + w\}$, the lemma follows by the above cases. ■

Adding the expressions in lemmas 2 and 3, we obtain:

$$\begin{aligned} N_q \cdot l_q &\leq \sum_{j=1}^{N_q - 1} \mathsf{On}(q, s(\tau_j), s(\tau_{j+1})) + \mathsf{On}(q, t_A - w, t_C) \\ &\leq \mathsf{On}(q, t_A - w, s(\tau_1)) + \sum_{j=1}^{N_q - 1} \mathsf{On}(q, s(\tau_j), s(\tau_{j+1})) + \mathsf{On}(q, t_A - w, t_A + w) \\ &= \mathsf{On}(q, t_A - w, s(\tau_{N_q})) + \mathsf{On}(q, t_A - w, t_A + w) \\ &\leq \mathsf{On}(q, t_A - w, t_B) + \mathsf{On}(q, t_A - w, t_A + w) \end{aligned}$$

Now summing this inequality over all pages $q \in [n]$,

$$\sum_{q=1}^{n} N_q \cdot l_q \leq \sum_{q=1}^{n} \mathsf{On}(q, t_A - w, t_B) + \sum_{q=1}^{n} \mathsf{On}(q, t_A - w, t_A + w) \leq t_B - t_A + 3w + 2, \tag{4.7}$$

where the last inequality follows from the fact that On is 1-speed. On the other hand, Rnd is always busy during $\mathcal{T}$: it is always broadcasting some tuple in $\bigcup_{q=1}^{n} \Pi_q$. Furthermore, since Rnd can broadcast an extra unit of page every $\lceil \frac{1}{\epsilon} \rceil$ time units, we obtain:

$$\sum_{q=1}^{n} N_q \cdot l_q \geq (1 + \epsilon/2) \cdot (t_B - t_A) - 3.$$

Combining this with equation (4.7), we have $t_B - t_A \leq \frac{6}{\epsilon} \cdot w + \frac{10}{\epsilon}$, which implies equation (4.6). Thus we obtain Theorem 1.

References

1. Bansal, N., Charikar, M., Khanna, S., Naor, J.: Approximating the average response time in broadcast scheduling. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 215–221. ACM, New York (2005) (electronic)
2. Bansal, N., Coppersmith, D., Sviridenko, M.: Improved approximation algorithms for broadcast scheduling. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 344–353. ACM, New York (2006)
3. Bansal, N., Krishnaswamy, R., Nagarajan, V.: Better scalable algorithms for broadcast scheduling. Technical Report CMU-CS-09-174, Carnegie Mellon University, Pittsburgh (2009)
4. Bartal, Y., Muthukrishnan, S.: Minimizing maximum response time in scheduling broadcasts. In: SODA 2000: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, pp. 558–559 (2000)

5. Chang, J., Erlebach, T., Gailis, R., Khuller, S.: Broadcast scheduling: algorithms and complexity. In: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 473–482. ACM, New York (2008)
6. Charikar, M., Khuller, S.: A robust maximum completion time measure for scheduling. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 324–333. ACM, New York (2006)
7. Chekuri, C., Im, S., Moseley, B.: Longest wait first and broadcast scheduling. In: WAOA 2009- Workshop on Approximation and Online Algorithms (2009)
8. Chekuri, C., Im, S., Moseley, B.: Minimizing maximum response time and delay factor in broadcast scheduling. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 444–455. Springer, Heidelberg (2009)
9. Chekuri, C., Moseley, B.: Online scheduling to minimize the maximum delay factor. In: SODA 2009: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, PA, USA, pp. 1116–1125 (2009)
10. Edmonds, J.: Scheduling in the dark. Theoret. Comput. Sci. 235(1), 109–141 (2000); Selected papers in honor of Manuel Blum, Hong Kong (1998)
11. Edmonds, J., Pruhs, K.: Multicast pull scheduling: when fairness is fine. Algorithmica 36(3), 315–330 (2003) (Online algorithms)
12. Edmonds, J., Pruhs, K.: A maiden analysis of longest wait first. ACM Trans. Algorithms 1(1), 14–32 (2005)
13. Edmonds, J., Pruhs, K.: Scalably scheduling processes with arbitrary speedup curves. In: SODA 2009: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, PA, USA, pp. 685–692 (2009)
14. Erlebach, T., Hall, A.: NP-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. J. Sched. 7(3), 223–241 (2004)
15. Fung, S.P.Y., Zheng, F., Chan, W.-T., Chin, F.Y.L., Poon, C.K., Wong, P.W.H.: Improved on-line broadcast scheduling with deadlines. J. Sched. 11(4), 299–308 (2008)
16. Gandhi, R., Khuller, S., Kim, Y.-A., Wan, Y.-C.: Algorithms for minimizing response time in broadcast scheduling. Algorithmica 38(4), 597–608 (2004)
17. Gandhi, R., Khuller, S., Parthasarathy, S., Srinivasan, A.: Dependent rounding and its applications to approximation algorithms. J. ACM 53(3), 324–360 (2006) (electronic)
18. Im, S., Moseley, B.: An online scalable algorithm for average flowtime in broadcast scheduling. In: SODA 2010: Twentyfirst Annual ACM-SIAM Symposium on Discrete Algorithms (2010)
19. Kalyanasundaram, B., Pruhs, K.: Speed is as powerful as clairvoyance. J. ACM 47(4), 617–643 (2000)
20. Kalyanasundaram, B., Pruhs, K.R., Velauthapillai, M.: Scheduling broadcasts in wireless networks. J. Sched. 4(6), 339–354 (2001)
21. Kim, J.-H., Chwa, K.-Y.: Scheduling broadcasts with deadlines. Theoret. Comput. Sci. 325(3), 479–488 (2004)
22. Pruhs, K., Sgall, J., Torng, E.: Online scheduling. In: Handbook of Scheduling: Algorithms, Models, and Performance Analysis (2004)
23. Robert, J., Schabanel, N.: Pull-based data broadcast with dependencies: be fair to users, not to items. In: SODA 2007: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, pp. 238–247 (2007)

Max-min Online Allocations with a Reordering Buffer

Leah Epstein[1], Asaf Levin[2,⋆], and Rob van Stee[3,⋆⋆]

[1] Department of Mathematics, University of Haifa, 31905 Haifa, Israel
lea@math.haifa.ac.il
[2] Faculty of Industrial Engineering and Management, The Technion, 32000 Haifa, Israel
levinas@ie.technion.ac.il.
[3] University of Karlsruhe, Department of Computer Science, 76128 Karlsruhe, Germany
vanstee@ira.uka.de

Abstract. We consider online scheduling so as to maximize the minimum load, using a reordering buffer which can store some of the jobs before they are assigned irrevocably to machines. For m identical machines, we show an upper bound of $H_{m-1}+1$ for a buffer of size $m-1$. A competitive ratio below H_m is not possible with any finite buffer size, and it requires a buffer of size $\tilde{\Omega}(m)$ to get a ratio of $O(\log m)$. For uniformly related machines, we show that a buffer of size $m+1$ is sufficient to get an approximation ratio of m, which is best possible for any finite sized buffer. Finally, for the restricted assignment model, we show lower bounds identical to those of uniformly related machines, but using different constructions. In addition, we design an algorithm of approximation ratio $O(m)$ which uses a finite sized buffer. We give tight bounds for two machines in all the three models.

These results sharply contrast to the (previously known) results which can be achieved without the usage of a reordering buffer, where it is not possible to get a ratio below an approximation ratio of m already for identical machines, and it is impossible to obtain an algorithm of finite approximation ratio in the other two models, even for $m=2$. Our results strengthen the previous conclusion that a reordering buffer is a powerful tool and it allows a significant decrease in the competitive ratio of online algorithms for scheduling problems. Another interesting aspect of our results is that our algorithm for identical machines imitates the behavior of the greedy algorithm on (a specific set of) related machines, whereas our algorithm for related machines completely ignores the speeds until the end, and then only uses the relative order of the speeds.

1 Introduction

Scheduling problems are most frequently described in a framework of assigning jobs to machines. There are various kinds of scheduling problems depending upon the properties of the machines, allowable assignments, and the cost criteria. Our goal is to study a natural model where the assignment of jobs is performed dynamically, in the sense that jobs are being considered one by one, and generally must be assigned in this order, while the information on future jobs is unknown. We consider parallel machines, and

⋆ Chaya fellow.

⋆⋆ Research performed at Max Planck Institute for Informatics, Germany, and supported by the German Research Foundation (DFG).

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 336–347, 2010.

a problem variant which possesses features of both offline and online scenarios. These are not offline problems, since the input arrives gradually, but they are not purely online either, since we allow partial reordering of the input.

More specifically, in this paper we study a variant called *scheduling with a reordering buffer*, where a buffer, which can store a fixed number of unassigned jobs, is available. Thus each new job must be either assigned to a machine or stored in the buffer (possibly in the case where the buffer is already full, the algorithm is forced to evict another job from the buffer, which must be assigned to a machine immediately). A job which is assigned to a machine, is assigned irrevocably to that machine.

In this paper, we are concerned with max-min allocations, that is, the goal is maximizing the minimum load. The concept of such allocations is related to the notion of fair division of resources [2]. Originally, the goal function was used for describing systems where the complete system relies on keeping all the machines productive for as long as possible, as the entire system fails even in a case that just one of the machines ceases to be active [18]. Additional motivations for the goal function come from issues of Quality of Service. From the networking aspect, this problem has applications to basic problems in network optimization such as fair bandwidth allocation. Consider pairs of terminal nodes that wish to communicate; we would like to allocate bandwidth to the connections in a way that no link unnecessarily suffers from starvation, and all links get a fair amount of resources. Another motivation is efficient routing of traffic. Consider parallel links between pairs of terminal nodes. Requests for shifting flow are assigned to the links. We are interested in having the loads of the links balanced, in the sense that each link should be assigned a reasonable amount of flow, compared to the other links. Yet another incentive to consider this goal function is congestion control by fair queuing. Consider a router that can serve m shifting requests at a time. The data pieces of various sizes, needing to be shifted, are arranged in m queues (each queue may have a different data rate), each pays a price which equals the delay that it causes in the waiting line. Our goal function ensures that no piece gets a "preferred treatment" and that they all get at least some amount of delay.

We are mainly interested in two models of parallel machines, namely identical machines [20] and uniformly related machines [3,7]. The input is a stream of jobs, of indices $1, 2, \ldots$, where the single attribute of each job j is its processing time, p_j, also known as its size. The goal is always to partition the jobs into m subsets, where each subset is to be executed on one specific machine, that is, there are m machines, of indices $\{1, 2, \ldots, m\}$, and each job is to be assigned to one of them. Online algorithms see the input jobs one at a time, and need to assign each job before becoming familiar with the remaining input. In the identical machines model, the completion time (also called load) of a machine is the total size of the jobs assigned to it, while for uniformly related machines (also known as related machines), each machine i has a speed s_i associated with it, and the completion time (or load) of a machine is the total size of jobs assigned to it, scaled by the speed of the machine. Without loss of generality, let the speeds be $s_1 \leq \cdots \leq s_m$.

An additional common variant considered here is *restricted assignment* [5]. The machines have identical speeds, though each job j is associated with a subset of the machines, $M_j \subseteq \{1, 2, \ldots, m\}$, and can be assigned only to a machine in M_j.

Notations. Throughout the paper we use the following notations. The size of the buffer is denoted by K. We use OPT to denote an optimal solution as well as its value or profit (i.e., the load of its least loaded machine). For an (offline or online) algorithm ALG we denote its value or profit by ALG as well. For a minimization problem, the value of a solution is called its *cost* and the approximation ratio of ALG is the infimum $\mathcal{R}$ such that for any input, $\text{ALG} \leq \mathcal{R} \cdot \text{OPT}$, whereas for a maximization problem (such as the problem studied here), the approximation ratio of ALG is the infimum $\mathcal{R}$ such that for any input, $\mathcal{R} \cdot \text{ALG} \geq \text{OPT}$ (note that we use numbers greater than 1 for approximation ratios for a maximization problem). If the approximation ratio of ALG is at most r, then we say that it is an r-approximation. Let H_t denote the harmonic series, that is, $H_t = \sum_{i=1}^{t} \frac{1}{t}$.

Related work. This problem (without a buffer) has been well studied (known by different names such as "machine covering" and "the Santa Claus problem") in the computer science literature (see e.g. [18,12,11,24,6,15,17,2]). For identical machines, it is known that any online algorithm for identical machines has an approximation ratio of at least m (the proof is folklore, see [24,4]), and this bound can be achieved using a greedy algorithm which assigns each job to the least loaded machine, as was shown by Woeginger [24]. Before the usage of our approach of incorporating a reordering buffer became popular, there were other attempts to overcome this bound. Randomized algorithms were studied in [4], where it was shown that the best approximation ratio which can be achieved using randomization is $\tilde{\Theta}(\sqrt{m})$. Several types of semi-online variants were considered, where one of the following was assumed: the largest processing time of any job is given in advance, the total size is given in advance, or both are given. It was shown that in these cases, the best approximation ratio remains $\Theta(m)$ [23,8]. Here we show that our approach allows us to reduce the approximation ratio much more significantly. It should be noted, that an additional, much stronger, semi-online variant was studied as well, where it is assumed that jobs arrive sorted by non-increasing processing time. In this variant, which is much closer to an offline problem than our model, the approximation ratio is at most $\frac{4}{3}$ [11,12].

For uniformly related machines, no algorithm with finite approximation ratio exists even for two machines [4]. The semi-online problem where jobs arrive sorted by non-increasing order, and the problem where the profit of an optimal algorithm is known in advance, admit m-approximations, which is best possible in both cases. If the two types of information are combined, a 2-approximation is known [4]. To the best of our knowledge, no positive results are known for the case of restricted assignment. It is known that no finite approximation ratio can be achieved for a purely online algorithm, even for two machines [9], and even in the model of hierarchical machines, where for every j, the set M_j is a prefix of the machine set.

In all variants of scheduling with a buffer studied in the past, a finite length buffer (usually of size $O(m)$) already allowed to achieve the best possible approximation ratio (for any finite sized buffer). Moreover, in almost all cases, the approximation ratio is significantly reduced, compared to the best possible purely online algorithm. We next survey the results for the min-max allocation problem (also known as the minimum makespan problem), whose goal is dual to our goal function, with a buffer.

Kellerer et al. [22] and Zhang [25] considered the case of two identical machines, and showed that a buffer of size 1 allows to achieve an approximation ratio of $\frac{4}{3}$, which is best possible. For m identical machines, Englert et al.[16] showed that a buffer of size $O(m)$ is sufficient. Their algorithm has the best possible approximation ratio for every value of m, while this ratio tends to 1.47 for large m. It is known that for online scheduling, no algorithm can have an approximation ratio smaller than 1.853 [1,19]. For the more general case of uniformly related machines, it was shown in [13] that for two machines, a buffer of size 2 is sufficient to achieve the best approximation ratio. In fact, for some speed ratios between the two machines, a buffer of size 1 is sufficient, while for some other speed ratios, a buffer of size 1 provably is not enough to achieve the best bound. Note that it was shown by [16] that a buffer of size $m-1$ reduces the approximation ratio for uniformly related machines below the lower bound of the case without a reordering buffer, specifically, in addition to the case of identical machines, Englert et al. [16] designed a 2-approximation algorithm for related machines, which uses a buffer of size $O(m)$. Whereas, without a buffer it is known that no algorithm can have an approximation ratio below 2.43 [7]. Finally, for the standard online scheduling problem in the restricted assignment model, there are tight bounds of $\Theta(\log m)$ on the approximation ratio [5]. The lower bound still holds if there is a buffer, since the construction of [5] holds for fractional assignment (where a job can be split arbitrarily among the machines that can run it); each job can be replaced by very small jobs, and so the usage of a buffer does not change the result.

The analogous questions for preemptive scheduling on identical machines were resolved in [14]. In this variant, jobs can be arbitrarily distributed among the machines, with the restriction that two parts of one job cannot be simultaneously processed on two machines. The best possible upper bound over all values of m is $\frac{4}{3}$, while for the case without a buffer, this value is approximately 1.58, as was shown by Chen et al. [10].

Our results. For identical machines, we design an algorithm which uses a buffer of size $m-1$, and has an approximation ratio of at most $H_{m-1}+1$. For $m=2$, we design a different $\frac{3}{2}$-approximation algorithm, which is optimal. We show a lower bound of H_m for any finite sized buffer, and in addition, we show that for a buffer size of $o(\frac{m}{\log m})$, an upper bound of $O(\log m)$ cannot be achieved. Interestingly, our algorithm IMITATE imitates the behavior of a greedy algorithm for uniformly related machines, with the speeds $\{1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{m}\}$.

For the cases of related machines and restricted assignment, we extend the known negative results of [4,9], and show that a buffer of size at most $m-2$ does not admit an algorithm of finite approximation ratio. For an arbitrary sized buffer, we show lower bounds of m on the approximation ratio of any algorithm. The proofs of the lower bounds in the two models are very different, but the results are similar. We complement these result by $O(m)$-approximation algorithms, which use a finite sized buffer. Specifically, for the related machines model we design an algorithm of approximation ratio m, which uses a buffer of size $m+1$, and an algorithm of approximation ratio below $2m-1$, which uses a buffer of size $m-1$. For two machines we design a different 2-approximation algorithm, which uses a buffer of size 1. For the restricted assignment model we present an algorithm of approximation ratio at most $2m$ which uses a buffer of size $O(m^2)$. For two machines, we give a simpler algorithm of approximation

ratio 2, which only requires a buffer of size 1. In contrast to the algorithm for identical machines, which attributes *phantom speeds* to the machines, the algorithm for related machines ignores the speeds during the main loop, and in the final loop, only uses the relative order of speeds, rather than their values.

Omitted proofs are deferred to the full version.

2 Identical Machines

We start with the most elementary case of identical machines. We first show limitations on the performance and the required buffer size of any algorithm.

2.1 Lower Bounds

In some of the lower bound proofs, the input sequence starts with a large number of very small jobs. In such a case, having a buffer is not meaningful, since almost all jobs must be assigned. The difficulty is in spreading these small jobs among the machines without knowing the sizes of future jobs. The next proof has some similarities with a lower bound for the preemptive variant [21].

Theorem 1. *For any buffer size K, no online algorithm can have an approximation ratio below H_m.*

Proof. We only give the construction here, and defer the proof to the full version. Let $\varepsilon > 0$ be a very small constant such that $\varepsilon = 1/N$ for some $N \in \mathbb{N}$ which is divisible by $m!$. The input contains N jobs of size ε, where $N >> K$. Let $b_1 \leq \cdots \leq b_m$ be the resulting loads for a given algorithm, after the last of these jobs has arrived. Since at most K of them remain in the buffer, we have $1 - K\varepsilon \leq \sum_{i=1}^{m} b_i \leq 1$.

Next, j jobs of size $1/(m-j)$ arrive for some $0 \leq j \leq m-1$. These j jobs are called *large jobs*. For this input, the optimal value is $\text{OPT}_j = 1/(m-j)$. □

The following proposition implies that we in fact need a buffer of size $\Omega(m/\log m)$ to get an approximation ratio which is logarithmic in m.

Proposition 1. *Let $f(m)$ be a function where $f(m) < m$ for all $m > 1$. For a buffer size $f(m)$, no algorithm can have an approximation ratio below $m/(f(m)+1)$.*

2.2 Algorithm

While the lower bound constructions are relatively simple, it is harder to see what an algorithm of sufficiently small approximation ratio should do. Intuitively, given the lower bound construction in the proof of Theorem 1, the machines should be relatively unbalanced. We achieve this by acting as if the machines have speeds (the so-called phantom speeds). We next define the algorithm IMITATE, which has a buffer of size $m-1$. IMITATE always keeps the largest $m-1$ jobs in the buffer.

ALGORITHM IMITATE. We store the first $m-1$ jobs in the buffer. Upon an arrival of a job, in the case that the buffer already contains $m-1$ jobs, consider those jobs and the new job. Let the size of the smallest job among these m jobs be X. A job of size

X will be assigned while the other jobs are stored in the buffer. The algorithm imitates the behavior of the so-called post-greedy algorithm for uniformly related machines. We define the phantom speed of the machine of index i to be $\frac{1}{i}$. Let L_i denote the current load of machine i. The next job is assigned to a machine which minimizes $i \cdot (L_i + X)$.

Upon termination of the input sequence, let $b_2 \leq \ldots \leq b_m$ be the sizes of jobs remaining in the buffer. (If there are only $j < m-1$ jobs in the buffer, then we let $b_i = 0$ for $2 \leq i \leq m-j$.) The job of size b_i is assigned to machine i (machine 1 does not get a job). We call this job *the final job* of machine i.

In what follows, we use ℓ_i to denote the load of machine i before the assignment of the final job, and L_i be the final load of machine i. That is, $L_i = \ell_i + b_i$ for $2 \leq i \leq m$ and $L_1 = \ell_1$. Note that if $b_2 = 0$, that is, the buffer contains less than $m-1$ jobs upon termination of the input sequence, then $\ell_i = 0$ for all $1 \leq i \leq m$.

To analyze the algorithm, we start with proving the following lemmas.

Lemma 1. *Let $1 \leq i \leq m$ and $2 \leq k \leq m$, $i \neq k$. Then $k \cdot L_k \geq i \cdot \ell_i$.*

Proof. If $\ell_i = 0$, then the claim trivially holds. Otherwise, let μ be the size of the last job ever assigned to machine i (neglecting the final job). Let λ_k be the load of machine k at that time in which μ was assigned. Then $k(\lambda_k + \mu) \geq i\ell_i$ by the post-greedy assignment rule. We have $L_k = \ell_k + b_k \geq \lambda_k + \mu$, using the properties $b_k \geq \mu$, since just before the assignment of the final jobs, the buffer contains the largest $m-1$ jobs in the input, and $\ell_k \geq \lambda_k$, since ℓ_k is the load of machine k just before the final jobs are assigned, while λ_k is the load of this machine at an earlier time. Hence $L_k \geq \lambda_k + \mu \geq \frac{i \cdot \ell_i}{k}$. □

Lemma 2. *For $i \geq 2$, $\ell_1 \geq (i-1)\ell_i$.*

Proof. If $\ell_i = 0$, then we are done. Otherwise let μ be the size of the last job ever assigned to machine i prior to the allocation of the final jobs. Let λ_1 be the load of machine 1 at the time when we assigned μ. Then $\lambda_1 + \mu \geq i\ell_i$ by the post-greedy assignment. Since $\ell_i \geq \mu$ and $\ell_1 \geq \lambda_1$, we get $\ell_1 \geq \lambda_1 \geq i\ell_i - \mu \geq (i-1)\ell_i$. □

We denote the total size of all jobs except for the final ones by Δ. That is, we let $\Delta = \sum_{i=1}^{m} \ell_i$.

Lemma 3. $(H_{m-1} + 1)\ell_1 \geq \Delta$.

Proof. By Lemma 2, we conclude that for all $i \geq 2$, $\ell_1 \geq (i-1)\ell_i$, and hence $\frac{\ell_1}{i-1} \geq \ell_i$. We sum up the last inequality for all $2 \leq i \leq m$, and we get $\sum_{i=2}^{m} \frac{\ell_1}{i-1} \geq \sum_{i=2}^{m} \ell_i = \Delta - \ell_1$. On the other hand, $\sum_{i=2}^{m} \frac{\ell_1}{i-1} = H_{m-1}\ell_1$, so $(H_{m-1} + 1)\ell_1 \geq \Delta$. □

Lemma 4. *For any $k > 1$, $(H_m - \frac{1}{k})kL_k \geq \Delta - \ell_k$.*

Proof. For all $1 \leq i \leq m$ such that $i \neq k$, we have that $\ell_i \leq \frac{k \cdot L_k}{i}$ by Lemma 1. Summing up for all $i \neq k$ we get $\Delta - \ell_k = \sum_{i \neq k} \ell_i \leq \sum_{i \neq k} \frac{k \cdot L_k}{i} = kL_k \sum_{i \neq k} \frac{1}{i} = kL_k(H_m - \frac{1}{k})$. □

Lemma 5. *For any $1 \leq k \leq m$, $\text{OPT} \leq \frac{1}{k}(\Delta + \sum_{j=2}^{k} b_j)$.*

Theorem 2. *The approximation ratio of* IMITATE *is at most $H_{m-1} + 1$.*

3 Uniformly Related Machines

3.1 Lower Bounds

Theorem 3. *For a buffer of size at most $m-2$, no algorithm can have a finite approximation ratio on related machines.*

Proof. Let the speed of machine i be S^{i-1} for some large $S \geq m+1$. Without loss of generality assume that the buffer has size $m-2$. The input sequence starts with $m-1$ jobs of sizes $S, S^2, \ldots, S^{m-1}$. At least one of these jobs cannot remain in the buffer. Let S^k be the size of the assigned job. Let j be the machine it is assigned to. If $j \leq k$, there is another job of size 1. Otherwise, another job of size S^m arrives.

In the first case, there is a machine in $\{k+1, \ldots, m\}$ which does not receive a job in $\{S^k, \ldots, S^{m-1}\}$, this machine has a profit of at most $S^{k-1}/S^k = 1/S$ (there are m jobs, so if a machine gets two jobs then ALG $= 0$). In this case, OPT $= 1$.

In the second case, if the machines $\{k+1, \ldots, m\}$ have all jobs $\{S^k, \ldots, S^m\}$, then machines $\{1, \ldots, k\}$ only have $k-1$ jobs and ALG $= 0$. Again, each machine must have one job exactly. Thus the machine having S^k has a profit of at most $S^k/S^k = 1$, while OPT $= S$. Letting $S \to \infty$, we get a ratio of ∞. □

Theorem 4. *For any buffer size K, no algorithm can have an approximation ratio below m for related machines.*

Proof. Again, we only give the construction here. Let the speed of machine i be S^{i-1} for some large $S \geq m+1$. The first phase is the same as for identical machines (see Lemma 1). Then in the second phase for some $1 \leq j \leq m$, the next jobs are of sizes $1/S^{j-1}, 1/S^{j-2}, \ldots, 1/S$ and $S, S^2, \ldots, S^{m-j}$ (if $j = 1$ then the first set is empty and if $j = m$ then the second set is empty). In the second phase there is one job of each size, so there are $m-1$ jobs. We have that the optimal value in this case is $\text{OPT}_j = 1/(S^{j-1})$ (put the job of size $1/S^i$ on machine $j-i$ for $i \neq 0$ and all small jobs on machine j; note that i can be negative). □

3.2 An Algorithm for m Related Machines

We next present an algorithm of approximation ratio m which uses a buffer of size $m+1$. In the full version, we design another algorithm of an approximation ratio slightly less than $2m-1$, which uses a buffer of size $m-1$. Our algorithms ignore the exact speeds, and only use their relative order.

Here we show that by using just two extra buffer positions compared to the minimum number of positions required to get an algorithm of finite approximation ratio, it is possible to get an optimal approximation ratio of m and hence save a factor of almost 2. In the case $m = 2$, the algorithm can be easily modified to keep only two jobs in the buffer, rather than three.

The algorithm works as follows. The first $m+1$ jobs are stored in the buffer. Upon arrival of a job, it is possibly swapped with a job of the buffer to maintain the property that the buffer contains the largest jobs seen so far.

The algorithm runs List Scheduling (LS) [20] on the machines, while ignoring the speeds. That is, a job is assigned to a minimally loaded machine, that is, a machine for which the total size of jobs assigned to it so far is minimum. Let the remaining jobs in the buffer when the input ends be $c_0 \leq \cdots \leq c_m$. We slightly abuse notation and let c_i denote also the size of job c_i. If the buffer contains at most $m-1$ jobs, then any assignment is optimal, since in this case, $\text{OPT} = 0$. If there are m jobs in the buffer, then assigning job c_i to machine i results in an optimal solution. The case that the buffer contains $m+1$ jobs is analyzed below (even if no other jobs were assigned).

The jobs $\{c_0, c_1, \ldots, c_m\}$ are assigned in one of the following $2m-1$ ways, depending on which option gives the largest profit.

1. Assign c_0 to the least loaded machine (in terms of the total size of jobs assigned to it), and c_i to machine i for all $i = 1, 2, \ldots, m$.
2. Assign the jobs as in the previous case, but move c_j to machine m for some $1 \leq j \leq m-1$.
3. Assign c_i to machine i for all $i = 1, 2, \ldots, m$. assign c_0 to a machine j for some $1 \leq j \leq m-1$.

In our analysis below, we show that there is always at least one assignment which shows that the approximation ratio is at most m.

Theorem 5. *The approximation ratio of this algorithm is at most m.*

Proof. We scale the input such that $\text{OPT} = 1$. Let T be the total size of all the jobs. Then $T \geq \sum_{i=1}^{m} s_i$. If $c_i \geq s_i/m$ for $i = 1, \ldots, m$, we are done. Else, let k be the maximum index for which $c_k < s_k/m$. We consider three cases.

Case 1: $k = m$ (i.e., $c_m < s_m/m$). We analyze only options where c_0 is assigned greedily to the least loaded machine. Let $a_1, \ldots, a_m$ be total sizes of jobs assigned to machines, neglecting $c_1, \ldots, c_m$ but not c_0. That is, since c_0 is assigned greedily, it is counted as part of some a_i.

If $a_m = \max_i a_i$, we analyze the first assignment option. Recall that T is the total size of all the jobs, that is $T = \sum_{i=1}^{m}(a_i + c_i)$. The load of machine m is

$$\frac{a_m + c_m}{s_m} = \frac{\max_i a_i + \max_i c_i}{s_m} \geq \frac{T/m}{s_m} \geq \frac{1}{m}\,.$$

For each other machine $i \leq m-1$, we have $a_i + c_i \geq a_m$. To see this, note that if $a_m = 0$ this clearly holds. Otherwise, let x be the size of the last job assigned to machine m before c_m. Due to the assignment rule $a_i \geq a_m - x$. Since all the jobs in the buffer are larger than all other jobsand since $c_0 \leq c_i$, we conclude that $x \leq c_i$ and the claim follows. Hence, $\sum_{j=1}^{m} a_j = T - \sum_{j=1}^{m} c_j \geq T - m \cdot \frac{s_m}{m} \geq s_i$, due to the definition of T, the property $c_i \leq c_m < \frac{s_m}{m}$ and $T \geq \sum_{j=1}^{m} s_j \geq s_i + s_m$.

Finally,

$$\frac{a_i + c_i}{s_i} \geq \frac{a_m}{s_i} \geq \frac{1}{m s_i} \sum_{j=1}^{m} a_j \geq \frac{1}{m}$$

holds since $a_i + c_i \geq a_m$, $a_m = \max_j a_j \geq \frac{1}{m}\sum_{j=1}^{m} a_j$, and $\sum_{j=1}^{m} a_j \geq s_i$.

If $a_m < \max_i a_i$, let $j = \arg\max_i a_i < m$. Consider the second type of assignment, for this specific value of j. As before, $\sum_{t=1}^{m} a_t \geq s_i$, $a_j = \max_t a_t \geq \frac{1}{m}\sum_{t=1}^{m} a_t$ and so $a_j \geq \frac{s_i}{m}$ for all $i < m$. Similarly to the previous proof, we can show $a_i + c_i \geq a_j$, so

$$\frac{a_i + c_i}{s_i} \geq \frac{a_j}{s_i} \geq \frac{1}{m} \text{ for } 1 \leq i < m, i \neq j.$$

For machine j, already $a_j/s_j \geq 1/m$. Finally, for machine m, $a_m + c_j \geq a_j \geq a_i$ (for all $1 \leq i \leq m$). We need to bound $(a_m + c_j + c_m)/s_m$ and we get $(a_m + c_j) + c_m \geq \max_i a_i + \max_i c_i \geq T/m \geq s_m/m$.

Case 2: $1 < k < m$. Consider the third assignment option, where c_0 is assigned to machine k. We now let a_i denote the total size of jobs assigned to machine i before the jobs $c_0, \ldots, c_m$ are assigned, so $T = \sum_{i=1}^{m} a_i + \sum_{i=0}^{m} c_i$.

For machines $k+1 \leq i \leq m$, we have $(a_i + c_i)/s_i \geq c_i/s_i \geq \frac{1}{m}$ and are done.

Let $j = \arg\max_i a_i$. We have $\sum_{i=0}^{k} c_i < (k+1) \cdot \frac{s_k}{m} \leq s_k$ since $k < m$. There are at least k machines of OPT that have no jobs in the set $\{c_{k+1}, \ldots, c_m\}$, so $\sum_{i=1}^{m} a_i + \sum_{i=0}^{k} c_i \geq s_1 + \cdots + s_k$, or $\sum_{i=1}^{m} a_i \geq s_{k-1}$ (using the property $k > 1$) and therefore $a_j \geq s_{k-1}/m$. As before, we have $a_i + c_i \geq a_j$ for any machine $1 \leq i \leq m$, so for $1 \leq i \leq k-1$, we find $(a_i + c_i)/s_i \geq a_j/s_{k-1} \geq 1/m$.

However, $a_k + c_0 \geq a_j$, since the assignment of the last job of machine j (excluding c_j) to machine k would have increased the total size of jobs assigned to it to at least a_j, and c_0 is no smaller than that job, so

$$a_k + c_0 + c_k \geq a_j + c_k \geq \frac{\sum_{i=1}^{m} a_i}{m} + \frac{\sum_{i=0}^{k} c_i}{k+1} \geq \frac{s_k}{m},$$

since $a_j \geq a_i$ for all $1 \leq i \leq m$, $c_k \geq c_i$ for all $0 \leq i \leq k$, and $k+1 \leq m$.

Case 3: $k = 1$. We have $c_i \geq s_i/m$ for $i = 2, \ldots, m$, so we only need to consider machine 1. Consider the first assignment, and use the notations a_i as in the first case. At least one machine of OPT has no jobs of $c_2, \ldots, c_m$, so $\sum_{i=1}^{m} a_i + c_1 \geq s_1 = \min_i s_i$. We have $a_1 + c_1 \geq a_i$ for $2 \leq i \leq m$, so $a_1 + c_1 \geq s_1/m$ and $(a_1 + c_1)/s_1 \geq 1/m$. □

4 Restricted Assignment

4.1 Lower Bounds

We first state the lower bounds which we show for the case of restricted assignment.

Theorem 6. *For a buffer of size at most $m - 2$, no algorithm can have a finite approximation ratio in the restricted assignment setting, that is, a buffer of size $\Omega(m)$ is required for a finite approximation ratio.*

Theorem 7. *For any finite-sized buffer, any online algorithm has an approximation ratio of at least m.*

4.2 An Algorithm for Restricted Assignment with a Finite-Sized Buffer

We now present an algorithm for restricted assignment. At each time, for every machine i, we keep the m largest jobs which can be processed on i, in the buffer. Every job which is not in this set (or a job which stops being in this set) is assigned in a greedy fashion to a least loaded machine which can process it. Thus, the required size of the buffer is m^2. Assume $\text{OPT} > 0$, so every machine has at least one job that can be assigned to it.

At the end of the input we assign the jobs from the buffer in an optimal way (i.e., we test all possible assignments and pick the best one). Note that it is possible to reduce the number of tested assignments while maintaining optimality. We will analyze a fixed assignment which depends on a (fixed) optimal solution OPT, which we define next.

Let S be the set of machines such that OPT has at least one job of size $\text{OPT}/(2m)$ assigned to it. Then, we would like to pick for every machine in S one job from the buffer of size at least $\text{OPT}/(2m)$, and assign it to that machine. Note that a machine having less than m jobs which it can process must belong to S, since in the optimal solution it has less than m jobs assigned to it, the largest of which has a size of at least $\text{OPT}/(m-1)$.

Lemma 6. *Such an assignment of jobs of size at least* $\text{OPT}/(2m)$ *is possible.*

Proof. Let j_i be the largest job (of size at least $\text{OPT}/(2m)$) assigned to machine i by OPT. If j_i is in the buffer, then we assign it to machine i. We do this for every $i \in S$. So far jobs were only assigned to their machines in OPT, so each job was assigned to at most one machine.

At the end of this process there might be additional machines in which we planned to assign to each such machine is not in the buffer. Therefore, the reason that for such a machine i we did not assign jobs is that there are more than m jobs which can be processed by i and which are larger than j_i. This means that i is a machine which initially (after the termination of the input) has m jobs in the buffer which can be assigned to it. At least one such job was not assigned before (since at most $m-1$ jobs from the buffer have been assigned), so we can assign it to i. Applying this procedure for one machine at a time, we will get an allocation of one job for each machine in S and such a job has size at least $\text{OPT}/(2m)$ as we required. □

After dealing with the set S, we continue to allocate one job to each machine not in S. We apply the following rule, for each machine $i \notin S$ we allocate to i the largest job in the buffer, which can be run on i, and which was not allocated before. We again assign a job to one machine at a time. We can always allocate a job from the buffer to i because initially there are at least m jobs which can be processed by i, and every machine is allocated exactly one job from the buffer.

Lemma 7. *If machine $i \notin S$ is allocated a job j, then for every job j' of size $p_{j'}$, which can be processed on machine i and is either still in the buffer after each machine received a job out of the buffer, or was allocated by the list scheduling algorithm before the input terminated, we have $p_j \geq p_{j'}$.*

Proof. Job j is larger than any job which is left in the buffer after every machine was allocated one job out of those left in the buffer (taking into account only the jobs which

can be processed by machine i). The jobs assigned greedily before the final step are no larger than j, since the buffer keeps the m largest jobs which i can receive. □

The jobs remaining in buffer, after each machine received a job out of the buffer, are assigned one by one, so that each job is assigned greedily to the least loaded machine that can process it. We say that a job j is small if it was allocated greedily (either at the earlier stage or after the input ends). Other jobs are called large. Therefore, the algorithm has allocated exactly one large job for each machine, and if OPT has assigned a job of size at least $\text{OPT}/(2m)$ to machine i, then the large job of i is of size at least $\text{OPT}/(2m)$.

Theorem 8. *Every machine is allocated a load of at least* $\text{OPT}/(2m)$.

Proof. A machine $i \in S$ is allocated a large job of size at least $\text{OPT}/(2m)$ as we discussed above. Hence, it suffices to consider a machine $i \notin S$ whose large job is of size less than $\text{OPT}/(2m)$. Fix such a machine i. For every machine $j \neq i$ we denote by C_j the set of jobs which OPT assigns to i and the algorithm assigns to j. Note that C_j may contain a large job, but in this case the large job is of size at most $\text{OPT}/(2m)$ (as otherwise $i \in S$ which is a contradiction).

We consider the total size of small jobs in C_j. Denote by x the last small job from the set C_j assigned to j. Note that by the greedy fashion in which jobs are assigned we conclude that if we discard x from C_j, the total size of remaining small jobs in C_j is at most the total size of small jobs assigned to i (as otherwise the algorithm would not assign x to j). Recall that the large job of i is at least as large as x, and hence we conclude that the total size of small jobs of C_j is at most the total assigned jobs to machine i. Summing up over all j we conclude that the total size of small jobs which OPT assigns to machine i and the algorithm assigns to other machines is at most $m-1$ times the load of machine i in the solution returned by the algorithm. The total size of large jobs which OPT assigns to machine i is at most $\text{OPT}/2$ (as each such job has size less than $\text{OPT}/(2m)$). Hence, the total size of small jobs which OPT assigns to machine i is at least OPT/2. Therefore, the algorithm assigns at least $\text{OPT}/(2m)$ to machine i, and the claim follows. □

References

1. Albers, S.: Better bounds for online scheduling. SIAM Journal on Computing 29(2), 459–473 (1999)
2. Asadpour, A., Saberi, A.: An approximation algorithm for max-min fair allocation of indivisible goods. In: Proc. 39th Symp. Theory of Computing (STOC), pp. 114–121 (2007)
3. Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., Waarts, O.: On-line load balancing with applications to machine scheduling and virtual circuit routing. Journal of the ACM 44(3), 486–504 (1997)
4. Azar, Y., Epstein, L.: On-line machine covering. In: Burkard, R.E., Woeginger, G.J. (eds.) ESA 1997. LNCS, vol. 1284, pp. 23–36. Springer, Heidelberg (1997)
5. Azar, Y., Naor, J., Rom, R.: The competitiveness of on-line assignments. Journal of Algorithms 18(2), 221–237 (1995)
6. Bansal, N., Sviridenko, M.: The Santa Claus problem. In: Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC), pp. 31–40 (2006)

7. Berman, P., Charikar, M., Karpinski, M.: On-line load balancing for related machines. Journal of Algorithms 35, 108–121 (2000)
8. Cai, S.-Y.: Semi-online machine covering. Asia-Pacific J. of Oper. Res. 24(3), 373–382 (2007)
9. Chassid, O., Epstein, L.: The hierarchical model for load balancing on two machines. Journal of Combinatorial Optimization 15(4), 305–314 (2008)
10. Chen, B., van Vliet, A., Woeginger, G.J.: An optimal algorithm for preemptive on-line scheduling. Operations Research Letters 18, 127–131 (1995)
11. Csirik, J., Kellerer, H., Woeginger, G.: The exact LPT-bound for maximizing the minimum completion time. Operations Research Letters 11, 281–287 (1992)
12. Deuermeyer, B.L., Friesen, D.K., Langston, M.A.: Scheduling to maximize the minimum processor finish time in a multiprocessor system. SIAM Journal on Discrete Mathematics 3(2), 190–196 (1982)
13. Dósa, G., Epstein, L.: Online scheduling with a buffer on related machines. Journal of Combinatorial Optimization. (to appear), doi:10.1007/s10878-008-9200-y
14. Dósa, G., Epstein, L.: Preemptive online scheduling with reordering. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 456–467. Springer, Heidelberg (2009)
15. Ebenlendr, T., Noga, J., Sgall, J., Woeginger, G.J.: A note on semi-online machine covering. In: Erlebach, T., Persinao, G. (eds.) WAOA 2005. LNCS, vol. 3879, pp. 110–118. Springer, Heidelberg (2006)
16. Englert, M., Özmen, D., Westermann, M.: The power of reordering for online minimum makespan scheduling. In: Proc. 48th Symp. Foundations of Computer Science (FOCS), pp. 603–612 (2008)
17. Epstein, L., Sgall, J.: Approximation schemes for scheduling on uniformly related and identical parallel machines. Algorithmica 39(1), 43–57 (2004)
18. Friesen, D.K., Deuermeyer, B.L.: Analysis of greedy solutions for a replacement part sequencing problem. Mathematics of Operations Reasearch 6(1), 74–87 (1981)
19. Gormley, T., Reingold, N., Torng, E., Westbrook, J.: Generating adversaries for request-answer games. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 564–565 (2000)
20. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell Sys. Tech. J. 45, 1563–1581 (1966)
21. Jiang, Y., Tan, Z., He, Y.: Preemptive machine covering on parallel machines. Journal of Combinatorial Optimization 10(4), 345–363 (2005)
22. Kellerer, H., Kotov, V., Speranza, M.G., Tuza, Z.: Semi online algorithms for the partition problem. Operations Research Letters 21, 235–242 (1997)
23. Tan, Z., Wu, Y.: Optimal semi-online algorithms for machine covering. Theoretical Computer Science 372(1), 69–80 (2007)
24. Woeginger, G.J.: A polynomial time approximation scheme for maximizing the minimum machine completion time. Operations Research Letters 20(4), 149–154 (1997)
25. Zhang, G.: A simple semi on-line algorithm for $P2//C_{\max}$ with a buffer. Information Processing Letters 61, 145–148 (1997)

Orientability of Random Hypergraphs and the Power of Multiple Choices

Nikolaos Fountoulakis and Konstantinos Panagiotou*

Max-Planck-Institute for Informatics
Saarbrücken, Germany
{fountoul,kpanagio}@mpi-inf.mpg.de

Abstract. A hypergraph $H = (V, E)$ is called s-orientable, if there is an assignment of each edge $e \in E$ to one of its vertices $v \in e$ such that no vertex is assigned more than s edges. Let $H_{n,m,k}$ be a hypergraph, drawn uniformly at random from the set of all k-uniform hypergraphs with n vertices and m edges. In this paper we establish the threshold for the 1-orientability of $H_{n,m,k}$ for all $k \geq 3$, i.e., we determine a critical quantity c_k^* such that with probability $1 - o(1)$ the graph $H_{n,cn,k}$ has a 1-orientation if $c < c_k^*$, but fails doing so if $c > c_k^*$.

We present two applications of this result that involve the paradigm of multiple choices. First, we show how it implies sharp load thresholds for cuckoo hash tables, where each element chooses k out of n locations. Particularly, for each $k \geq 3$ we prove that with probability $1 - o(1)$ the maximum number of elements that can be hashed is $(1 - o(1))c_k^* n$, and more items prevent the successful allocation. Second, we study random graph processes, where in each step we have the choice among any edge connecting k random vertices. Here we show the existence of a phase transition for avoiding a giant connected component, and quantify precisely the dependence on k.

1 Introduction

A hypergraph is called *s-orientable* if for each edge we can select one of its vertices, so that all vertices are selected at most s times. This definition generalizes the classical notion of orientability of graphs, where we want to orient the edges under the condition that no vertex has in-degree larger than s. In this paper we focus on the property of 1-orientability of random hypergraphs. In particular, we consider random k-uniform hypergraphs, for $k \geq 3$, with n vertices and $m = \lfloor cn \rfloor$ edges. Our main result establishes the existence of a critical density c_k^*, such that when c crosses this value the probability that the random hypergraph is 1-orientable drops abruptly from $1 - o(1)$ to $o(1)$, as the number of vertices n grows.

Similar sharp threshold behaviour has been established in the case of random graphs for the property of s-orientability for any $s \geq 1$ by Fernholz and Ramachandran in [13]. However, these critical densities for k-uniform random

* This author is supported by the Humboldt Foundation.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 348–359, 2010.

hypergraphs with $k \geq 3$ are only known for very large s – see the recent paper by Gao and Wormald [16]. Let $H_{n,\lfloor cn \rfloor,k}$ denote a k-uniform hypergraph, drawn uniformly at random for the set of k-uniform hypergraphs with vertex set $V_n := \{1, \ldots, n\}$ and $\lfloor cn \rfloor$ edges. We show the following result.

Theorem 1. *For any integer $k \geq 3$ let ξ^* be the unique solution of the equation*

$$k = \frac{\xi^*(e^{\xi^*} - 1)}{e^{\xi^*} - 1 - \xi^*}, \tag{1.1}$$

and set $c_k^* = \frac{\xi^*}{k(1-e^{-\xi^*})^{k-1}}$. Then the following holds with probability $1 - o(1)$.

1. *If $c < c_k^*$, then $H_{n,\lfloor cn \rfloor,k}$ is 1-orientable.*
2. *If $c > c_k^*$, then $H_{n,\lfloor cn \rfloor,k}$ is not 1-orientable.*

Numerically we obtain for example that $c_3^* \doteq 0.917, c_4^* \doteq 0.976$ and $c_5^* \doteq 0.992$, where "$\doteq$" indicates that the values are truncated to the last digit shown. Moreover, a simple calculation reveals that $c_k^* = 1 - e^{-k} + o(e^{-k})$ for $k \to \infty$.

In the remainder we give two applications of the above theorem in the context of computer science as well as in discrete mathematics.

Cuckoo Hashing. The first application regards the technique of cuckoo hashing, which was introduced by Pagh and Rodler [21]. The general setting considered here is a slight variation of it, as defined by Fotakis, Pagh, Sanders and Spirakis in [14]. We are given a table with n locations, and we assume that each location can hold only one item. Further generalizations where two or more items can be stored have also been studied, see e.g. [10,5,13], but we will not treat those cases. Moreover, there is a set $\mathcal{I}$ of items, such that each $x \in \mathcal{I}$ is assigned $k \geq 2$ random distinct locations $\ell_1(x), \ldots, \ell_k(x)$. This assumption may be somehow idealized, as exponentially many bits would be needed to store such fully random sets of locations. However, there is theoretical evidence that even "simple" hash functions can be sufficient in practice, provided that the underlying data stream fulfills certain natural conditions; we refer the reader to the papers [19] by Mitzenmacher and Vadhan and [9] by Dietzfelbinger and Schellbach, and the references therein.

A natural question in cuckoo hashing is the following. As the number of available items increases, it becomes more and more unlikely that all of them can be inserted into the table such that each item is assigned to one of its k desired locations. In other words, if $|\mathcal{I}|$ is "small", then with high probability, i.e., with probability arbitrarily close to 1 when n becomes large, there is such an assignment to the locations in the table that respects the k choices of the items. On the other hand, if $|\mathcal{I}|$ becomes "large", then such an assignment does not exist with high probability (trivially, this happens at the latest when $n + 1$ items are available). The important question is whether there is a critical size for $\mathcal{I}$ where the probability for the existence of a valid assignment drops instantaneously in the limiting case from 1 to 0, i.e., whether there is a *load threshold* for cuckoo hashing.

Cuckoo hashing can be modeled with the use of random hypergraphs: the n vertices represent the locations, and each available item is a hyperedge that contains k vertices. Let us become a little more precise. Let $H^*_{n,m,k}$ denote a hypergraph that is drawn uniformly at random from the set of k-uniform multigraphs with n vertices and m edges. Then, an instance of $H^*_{n,m,k}$ corresponds precisely to a cuckoo hashing scenario, where the table consists of n locations, there are m items in total, and each item chooses k random locations as specified above.

The case of $k = 2$ choices is theoretically well-understood. As already mentioned, there is a natural correspondence to random graphs, which makes the analysis tractable; we refer the reader to [7] and [11] for further details.

Note that $H^*_{n,m,k}$ may have multiple edges, which happens when two items choose the same k locations. However, an easy calculation shows that the expected number of multiple edges is $o(1)$, and thus $H^*_{n,m,k}$ should be very similar to $H_{n,m,k}$ in a probabilistic sense.

Proposition 1. *Let $\mathcal{P}$ be a property of simple hypergraphs. Then, for any $c > 0$ and $k \geq 3$*

$$\mathbb{P}\left(H_{n,\lfloor cn\rfloor,k} \in \mathcal{P}\right) = \mathbb{P}\left(H^*_{n,\lfloor cn\rfloor,k} \in \mathcal{P}\right) + o(1).$$

Now, it can be readily seen that the allocation of $\lfloor cn\rfloor$ random items is possible in a cuckoo hash table if and only if $H^*_{n,\lfloor cn\rfloor,k}$ is 1-orientable. Thus the following is now immediate from Theorem 1 and Proposition 1.

Theorem 2. *The load threshold for cuckoo hashing with $k \geq 3$ choices is c^*_k.*

The result of Theorem 2 for $k \geq 4$ was obtained independently by Frieze and Melsted [15] using a different approach. Namely, they reduced the problem into finding a perfect matching on a suitable random bipartite graph, where they analyzed very precisely the Karp-Sipser algorithm. A different approach to the result of Theorem 2 was also followed by Dietzfelbinger, Goerdt, Mitzenmacher, Montanari, Pagh and Rink [8], who reduced it to the XORSAT problem.

Avoidance of Giants. The second application of our analysis has to do with the problem of the avoidance of giant connected components in random graphs. If one chooses $\lfloor cn\rfloor$ random edges on a set of n vertices, the celebrated result of Erdős and Rényi [12] states that $1/2$ is the critical value of c above which a *giant component* emerges with probability $1-o(1)$. More specifically, if $c < 1/2$ is fixed, then with probability $1 - o(1)$ all components contain at most $C \log n$ vertices, for some C depending on c. But if $c > 1/2$, then there exists $\gamma = \gamma(c) > 0$ such that with probability $1 - o(1)$ there is a unique component with γn vertices, whereas every other component has no more than $C' \log n$ vertices for some $C' = C'(c) > 0$. In this case, the unique largest component is called the *giant component*. D. Achlioptas posed the following question: if one is presented with $\lfloor cn\rfloor$ random pairs of edges on a set of n vertices, is it possible to choose an edge from each pair so that with probability $1 - o(1)$ no giant component exists even for values of c which are larger than $1/2$? In fact, originally an online version

of this question was posed: if the $\lfloor cn \rfloor$ random pairs appear sequentially and the choice of an edge from each pair is made without knowledge of the pairs that have not been presented yet, is there a method of choice that avoids the emergence of a giant component when $c > 1/2$? Such an online process is called an *Achlioptas process*. Bohman and Frieze [1] were the first to give explicitly such a process which is able to avoid the emergence of a giant component for $c < 0.535$. Subsequently, Spencer and Wormald [22] pushed this even further to 0.83. Moreover, Bohman, Frieze and Wormald [3] showed that an upper bound for c is 0.96445.

Here we deal with the offline version of this problem. That is, we are initially given $\lfloor cn \rfloor$ random sets of edges $A_1, \ldots, A_{\lfloor cn \rfloor}$. The question is whether with probability $1 - o(1)$ *every* set of edges E that intersects each A_i induces a giant connected component. Bohman and Kim [4] established a critical value for c in the case where the A_i's contain two independent random edges each. Here we generalize this question to the case where instead of pairs of edges the A_i's are sets of vertices of size k, and we are allowed to choose any of its induced edges.

Theorem 3. *Let $A_1, \ldots, A_{\lfloor cn \rfloor}$ be a sequence of subsets of V_n of size $k \geq 3$ chosen independently and uniformly at random. Then the following hold with probability $1 - o(1)$:*

1. *If $c > c_k^*$, then there exists $\delta > 0$ so that for every collection of edges E on V_n such that $E \cap A_i \neq \emptyset$, for all $i = 1, \ldots, \lfloor cn \rfloor$, the set E induces a connected component with at least δn vertices.*
2. *If $c < c_k^*$, then there exists a collection of edges E on V_n with $E \cap A_i \neq \emptyset$, for all $i = 1, \ldots, \lfloor cn \rfloor$, such that all connected components that the set E induces have $o(n)$ vertices.*

This theorem follows with some additional work from Theorems 4 and 5 stated below and imply our main result, i.e., Theorem 1. In short, these theorems state that c_k^* is the critical value for the emergence of a subgraph of $H_{n,\lfloor cn \rfloor,k}$ that contains more edges than vertices. As we will see in the next section, this property implies also the 1-orientability of $H_{n,\lfloor cn \rfloor,k}$.

2 Proof of the Main Result

In what follows we will be referring to a hyperedge of size k as a *k-edge* and we will be calling a hypergraph where all of its hyperedges are of size k a *k-graph*.

Suppose now that we are given a k-graph H with n vertices and m edges. Note that an assignment of the edge to the set of vertices such that every edge gets assigned to one of its vertices is possible, if the following condition is satisfied: every one of the induced subgraphs of H has less k-edges than vertices. We call the ratio of the number of edges of a k-graph over its number of vertices the *density* of this k-graph.

It turns out that this necessary condition is also sufficient – see the proof of Theorem 1 below. In other words, the crucial parameter that determines

whether an appropriate assignment of the edges to the vertices is possible is the maximal density of an induced subgraph. The next theorem says that if the number of items exceeds $c_k^* n$, then there is a subgraph with density > 1. Before we state it, let us introduce some additional notation. We define the *core* of a hypergraph H to be the maximum subgraph of H that has minimum degree at least 2; if such a subgraph does not exist then we say that the core *is empty*. The core of random hypergraphs and its structural characteristics have been studied quite extensively in recent years – see for example the papers by Cooper [6] or Molloy [20].

Theorem 4. *Let c_k^* be defined as in Theorem 1. If $c > c_k^*$, then with probability $1 - o(1)$ the core of $H_{n,cn,k}$ has density greater than 1.*

This theorem is not very difficult to prove, given the results in [6] and [18]. However, it requires some technical work, which is accomplished in Section 3. The heart of this paper is devoted to the "subcritical" case, where we show that the above result is essentially tight.

Theorem 5. *Let c_k^* be defined as in Theorem 1. If $c < c_k^*$, then with probability $1 - o(1)$ all subgraphs of $H_{n,cn,k}$ have density smaller than 1.*

The proof of this theorem is technically more challenging and it is spread over the remaining sections. With Theorems 4 and 5 at hand we are able to give the proof of Theorem 1.

Proof (Theorem 1). Given a hypergraph H with n vertices and m edges, let us construct an auxiliary bipartite graph $B = (I, L; E)$, where I represents the m edges, L represents the n vertices, and $\{i, \ell\} \in E$ if ℓ is one of the k vertices of edge i. Note that it is possible to assign all edges to vertices such that each edge is assigned to one of its vertices if and only if there is a matching in B that covers all vertices in I. By Hall's Theorem such a matching exists if and only if for all $I' \subseteq I$ we have that $|I'| \leq |\Gamma(I')|$, where $\Gamma(I')$ denotes the set of neighbors of the vertices in I' inside L.

As a next step, let us describe more precisely the set $\Gamma(I')$. For a random hypergraph $H_{n,m,k}$, the set $\Gamma(I')$ is precisely the set of vertices that are contained in the hyperedges that belong to I'. So, if $c < c_k^*$, Theorem 5 guarantees that with high probability for all I' we have $|\Gamma(I')| > |I'|$, and therefore a matching exists. On the other hand, if $c > c_k^*$, then there is with high probability an I' such that $|\Gamma(I')| < |I'|$; choose for example I' to be the set of items that correspond to the edges in the core of $H_{n,m,k}$. Hence a matching does not exist in this case, and the proof is completed.

3 Properties of Random k-Graphs

The aim of this section is to determine the value c_k^* and prove Theorem 4. Moreover, we will introduce some known facts and tools that will turn out to be very useful in the study of random hypergraphs, and will be used later on in the proof of Theorem 5 as well.

(More) Models of random k-graphs. For the sake of convenience we will carry out our calculations in the $H_{n,p,k}$ model of random k-graphs. This is the "higher-dimensional" analogue of the well-studied $G_{n,p}$ model, where each possible (2-)edge is included independently with probability p. More precisely, given $n \geq k$ vertices we obtain $H_{n,p,k}$ by including each k-tuple of vertices with probability p, independently of every other k-tuple.

Standard arguments show that if we adjust p suitably, then the $H_{n,p,k}$ model is essentially equivalent to the $H_{n,cn,k}$ model. Roughly speaking, if we set $p = ck/\binom{n-1}{k-1}$, then $H_{n,p,k}$ is expected to have $p\binom{n}{k} = cn$ edges. In fact, much more is true. Let $\mathcal{P}$ be a *convex* hypergraph property, that is, whenever we have three hypergraphs on the same vertex set H_1, H_2, H_3 such that $H_1 \subseteq H_2 \subseteq H_3$ and $H_1, H_3 \in \mathcal{P}$, then $H_2 \in \mathcal{P}$ as well. (We also assume that $\mathcal{P}$ is closed under automorphisms.) Clearly any monotone property is also convex and, therefore, in particular the properties examined in Theorem 5. The following proposition is a generalization of Proposition 1.15 from [17, p.16] and its proof is very similar to the proof of that – so we omit it.

Proposition 2. *Let $\mathcal{P}$ be a convex property of hypergraphs, and let $p = ck/\binom{n-1}{k-1}$, where $c > 0$. If $\mathbb{P}(H_{n,p,k} \in \mathcal{P}) \to 1$ as $n \to \infty$, then $\mathbb{P}(H_{n,cn,k} \in \mathcal{P}) \to 1$ as well.*

Working on the core of $H_{n,p,k}$ – the Poisson cloning model. Recall that the core of a hypergraph is its maximum subgraph that has minimum degree at least 2. At this point we introduce the main tool for our analysis of the core of $H_{n,p,k}$. This is the so-called *Poisson cloning model* that was introduced by Kim [18] and was used for a variety of problems. Our treatment here was inspired by the analysis of Bohman and Kim [4] in the context of Achlioptas processes.

The Poisson cloning model $\widetilde{H}_{n,p,k}$ for k-graphs with n vertices and parameter $p \in [0,1]$ is defined as follows. Consider a set of n vertices V_n and consider also a family, indexed by this set, of i.i.d. Poisson random variables with parameter $\lambda := p\binom{n-1}{k-1}$. For each $v \in V_n$ let $d(v)$ denote the corresponding random variable from this family. Then $\widetilde{H}_{n,p,k}$ is constructed in three steps as follows. First, for every $v \in V_n$ the degree of v is a random variable and equals $d(v)$. Second, for each such v we generate $d(v)$ copies, which we call *v-clones* or simply *clones*, and choose uniformly at random a matching from all perfect k-matchings on the set of all clones. Note that such a matching may not exist – in this case we choose a random matching that leaves less than k clones unmatched. Finally, we generate $\widetilde{H}_{n,p,k}$ by contracting the clones to vertices, i.e., by projecting the clones of v onto v itself for every $v \in V_n$.

Note that the last two steps in the above procedure are together equivalent to the *configuration model* $H_{\mathbf{d},k}$ for random hypergraphs with degree sequence $\mathbf{d} = (d_1, \ldots, d_n)$. In other words, $H_{\mathbf{d},k}$ is a random multigraph where the ith vertex has degree d_i.

The following statement is implied by [18, Theorem 1.1], and says that the study of $H_{n,p,k}$ may be reduced to the study of the Poisson cloning model.

Theorem 6. *If $\mathbb{P}\left(\widetilde{H}_{n,p,k} \in \mathcal{P}\right) \to 0$ as $n \to \infty$, then $\mathbb{P}(H_{n,p,k} \in \mathcal{P}) \to 0$ as well.*

The next result that we shall exploit gives a precise description of the core of $\tilde{H}_{n,p,k}$. Particularly, Theorem 6.2 in [18] implies the following statement, where we write "$x \pm y$" for the interval of numbers $(x-y, x+y)$.

Theorem 7. *Let* $\lambda_2 := \min_{x>0} \frac{x}{(1-e^{-x})^{k-1}}$. *Moreover, let* $\bar{x}$ *be the largest solution of the equation* $x = (1-e^{-xck})^{k-1}$, *and set* $\xi := \bar{x}ck$. *Assume that* $ck = p\binom{n-1}{k-1} = \lambda_2 + \sigma$, *where* $\sigma > 0$ *is fixed. Then, for any* $0 < \delta < 1$ *the following is true with probability* $1 - o(1)$. *If* $\tilde{N}_2$ *denotes the number of vertices in the core of* $\tilde{H}_{n,p,k}$, *then*

$$\tilde{N}_2 = (1 - e^{-\xi} - \xi e^{-\xi})n \pm \delta n.$$

Furthermore, the core itself is distributed like the Poisson cloning model on $\tilde{N}_2$ *vertices, where the Poisson random variables are* conditioned on being at least two *and have parameter* $\Lambda_{c,k}$, *where* $\Lambda_{c,k} = \xi + \beta$, *for some* $|\beta| \leq \delta$.

In what follows, we say that a random variable is a 2-truncated Poisson variable, if it is distributed like a Poisson variable, conditioned on being at least two. We immediately obtain by Chebyshev's inequality the following corollary.

Corollary 1. *Let* N_2 *and* M_2 *denote the number of vertices and edges in the core of* $H_{n,p,k}$. *Then, for any* $0 < \delta < 1$, *with probability* $1 - o(1)$,

$$N_2 = (1 - e^{-\xi} - \xi e^{-\xi})n \pm \delta n \quad \text{and} \quad M_2 = \frac{1}{k}\xi(1 - e^{-\xi})n \pm \delta n,$$

where $\xi = \bar{x}ck$ *and* $\bar{x}$ *is the largest solution of the equation* $x = (1-e^{-xck})^{k-1}$. *The same is true for the quantities* $\tilde{N}_2$ *and* $\tilde{M}_2$ *of* $\tilde{H}_{n,p,k}$.

3.1 Proof of Theorem 4 and the Value of c_k^*

In this section we will prove Theorem 4, i.e., we will show that the core of $H_{n,p,k}$ has density at least one if $p = ck/\binom{n-1}{k-1}$ and $c > c_k^*$. Let $0 < \delta < 1$, and denote by N_2 and M_2 the number of vertices and edges in the core of $H_{n,p,k}$. By applying Corollary 1 we obtain that with probability $1 - o(1)$

$$N_2 = (1 - e^{-\xi} - \xi e^{-\xi})n \pm \delta n \quad \text{and} \quad M_2 = \frac{1}{k}\xi(1 - e^{-\xi})n \pm \delta n,$$

where $\xi = \bar{x}ck$ and $\bar{x}$ is the largest solution of the equation $x = (1-e^{-xck})^{k-1}$. The value of c_k^* is then obtained by taking $N_2 = M_2$, and ignoring the additive error terms. The above values imply that the critical ξ^* is given by the equation

$$1 - e^{-\xi^*} - \xi^* e^{-\xi^*} = \frac{\xi^*}{k}(1 - e^{-\xi^*}) \implies k = \frac{\xi^*(1-e^{-\xi^*})}{1 - e^{-\xi^*} - \xi^* e^{-\xi^*}}.$$

This is precisely (1.1). So, k determines ξ^* and $\bar{x}$ satisfies $\bar{x} = (1-e^{-\bar{x}ck})^{k-1} = (1-e^{-\xi^*})^{k-1}$. Therefore, the critical density is

$$c_k^* := \frac{1}{k}\frac{\xi^*}{(1-e^{-\xi^*})^{k-1}}. \tag{3.1}$$

Proof (Theorem 4). The above calculations imply that for any $0 < \delta < 1$ with probability $1 - o(1)$

$$\frac{M_2}{N_2} = \frac{1}{k} \frac{\xi(1 - e^{-\xi})}{1 - e^{-\xi} - \xi e^{-\xi}} \pm 2\delta. \tag{3.2}$$

Moreover, if $c = c_k^*$, then $M_2/N_2 = 1 \pm 2\delta$. To complete the proof it is therefore sufficient to show that M_2/N_2 is an increasing function of c. Note that the ratio $\frac{\xi(1-e^{-\xi})}{1-e^{-\xi}-\xi e^{-\xi}}$ is the expected value of a 2-truncated Poisson random variable with parameter ξ, which is known (and easily seen) to be increasing in ξ. Recall that $\xi = \bar{x}ck$. The proof is now complete through the first part of the following claim.

Claim. The quantity $\xi = \bar{x}ck$ is increasing with respect to c. So, with probability $1 - o(1)$

$$\frac{M_2}{N_2} < 1 \quad \text{, if } c < c_k^* \qquad \text{and} \qquad \frac{M_2}{N_2} > 1 \quad \text{, if } c > c_k^*.$$

4 Proof of Theorem 5

Let us begin with introducing some notation. For a hypergraph H we will denote by V_H its vertex set and by E_H its set of edges. Additionally, v_H and e_H shall denote the number of elements in the corresponding sets. For $U \subset V_H$ we denote by v_U, e_U the number of vertices in U and the number of edges joining vertices only in U. Finally, d_U is the total degree in U, i.e., the sum of the degrees in H of all vertices in U.

We say that a subset U of the vertex set of a hypergraph is 1-dense, if it has density at least one, i.e., the subgraph induced by U has at least as many edges as vertices. Working towards the proof of Theorem 5, we begin with showing that whenever $c < 1$, $H_{n,cn,k}$ does not contain large 1-dense subsets, for any $k \geq 3$. In particular, we will first argue about sets with no more than $0.7n$ vertices; for them it is sufficient to use a first moment argument that is based on rough counting. For larger sets of vertices we need more sophisticated arguments regarding the structure of the core of $H_{n,cn,k}$.

The following statement deals with the case $k \geq 5$; is not best possible, but it suffices for our purposes.

Lemma 1. *Let $k \geq 5, c < 1$. Then $H_{n,cn,k}$ contains no 1-dense subset with less than $0.7n$ vertices with probability $1 - o(1)$.*

Proof. The probability that an edge of $H_{n,cn,k}$ is contained completely in a subset U of the vertex set is $\binom{|U|}{k}/\binom{n}{k} \leq (\frac{|U|}{n})^k$. Let $\frac{k}{n} \leq u \leq 0.7$. Then

$$\mathbb{P}\left(\exists \text{ 1-dense subset with } un \text{ vertices}\right) \leq \binom{n}{un} \cdot \binom{cn}{un} u^{kun} \leq e^{n(2H(u)+ku\ln u)},$$

where $H(x) = -x\ln x - (1-x)\ln x$ denotes the entropy function. Note that the second derivative of the exponent in the expression above is $\frac{k-2+kx}{x(1-x)}$, which

is positive for $x \in (0,1)$. Hence the exponent is convex, implying that it is maximized either at $u = k/n$ or at $u = 0.7$. Note that

$$2H(0.7) + k0.7\ln(0.7) \leq 2H(0.7) + 5 \cdot 0.7\ln(0.7) \leq -0.02$$

and that

$$2H\left(\frac{k}{n}\right) + \frac{k^2}{n}\ln\left(\frac{k}{n}\right) = -\frac{(k^2-2k)\ln n}{n} + O\left(\frac{1}{n}\right).$$

So, the minimum is obtained at $u = k/n$, and we conclude the proof with

$$\mathbb{P}\left(\exists \text{ 1-dense subset with } \leq 0.7n \text{ vertices}\right) = \sum_{k/n \leq u \leq 0.7} O(n^{-k^2+2k}) = O(n^{-14}).$$

For the case $k = 3$ as well as for the rest of our analysis we need to exploit more sophisticated properties of 1-dense sets. In particular, we will use the following statement, which was observed by Bohman and Kim [4]. We present a proof for the sake of completeness.

Proposition 3. *Let H be a k-graph with density < 1 and let U be an inclusion maximal 1-dense subset of V_H. Then $e_U = v_U$ and all edges $e \in E_H$ satisfy $|e \cap U| \neq k-1$.*

Proof. If $e_U > v_U$, then let $U' = U \cup \{v\}$, where v is any vertex in $V_H \setminus U$. Note that such a vertex always exists, as $U \neq V_H$. Moreover, denote by d the degree of v in U, i.e., the number of edges in H that contain v and all other vertices only from U. Then

$$\frac{e_{U'}}{v_{U'}} = \frac{e_U + d}{v_U + 1} \geq \frac{e_U}{v_U + 1} \geq 1,$$

which contradicts the maximality of U. Similarly, if there was an edge e such that $|e \cap U| = k-1$, then we could construct a larger 1-dense subset of V_H by adding the vertex in $e \setminus U$ to U.

Motivated by the above proposition, we use the following stronger claim for random 3-uniform hypergraphs.

Lemma 2. *Let H be a 3-graph, and call a set $U \subset V_H$* bad *if $e_U = |U|$ and $\forall e \in E_H : |e \cap U| \neq 2$. Then, for any $c \leq 0.95$*

$$\mathbb{P}\left(H_{n,cn,3} \text{ contains a bad subset } U \text{ with } \leq n/2 \text{ vertices}\right) = o(1).$$

To argue about larger sets and, thus, complete the proof of Theorem 5, it suffices to focus our analysis on the core of $H_{n,cn,k}$. Indeed, suppose that $H_{n,cn,k}$ contains a 1-dense subset of vertices, and let U be such a minimal one (with respect to the number of vertices). Then each edge in the subgraph induced by U contains at least two vertices of U, as otherwise we could remove a vertex of degree one or zero in order to obtain a 1-dense set $U' \subset U$. This implies that the core of $H_{n,cn,k}$ contains all minimal 1-dense subsets. Therefore it suffices to focus on

the analysis of the core of $\widetilde{H}_{n,p,k}$ and show that with probability $1-o(1)$ it does not contain any 1-dense subset – by Theorem 6 this is implied for $H_{n,cn,k}$ too.

We will work with some c which is slightly below c_k^*. Let $\delta > 0$. As ξ is increasing with respect to c (cf. Claim 3.1), there exists a $\gamma = \gamma(\delta) > 0$ such that $c = c_k^* - \gamma$ and $\xi = \xi^* - \delta$, where ξ^* is the unique solution of $k = \frac{\xi^*(e^{\xi^*}-1)}{e^{\xi^*}-1-\xi^*}$. Let

$$n_2 = n(1 - e^{-\xi} - \xi e^{-\xi}) \text{ and } m_2 = (1 - e_k\delta)n_2,$$

where e_k is given by $\frac{\xi(e^{\xi}-1)}{e^{\xi}-\xi-1} = k(1-e_k\delta+\Theta(\delta^2))$. Note that Corollary 1 implies that $N_2 = n_2 \pm \delta^2 n$, using δ^2 instead of δ. Moreover, setting δ^3 instead of δ in (3.2) and taking Taylor's expansion around ξ^* imply that, with probability $1-o(1)$,

$$M_2 = m_2 \pm \delta^2 n. \tag{4.1}$$

Now, for $\beta, q \in [0,1]$ let $X_{q,\beta}^{(1)} = X_{q,\beta}^{(1)}(\mathrm{C})$ denote the number of subsets of the core C of $\widetilde{H}_{n,p,k}$ with $\lfloor \beta N_2 \rfloor$ vertices and total degree $\lfloor q \cdot kM_2 \rfloor$ that are maximal 1-dense. The main tool in our proofs is the following sharp bound.

Lemma 3. *Let T_z be the unique solution of $z = \frac{T_z(1-e^{-T_z})}{1-e^{-T_z}-T_z e^{-T_z}}$, where $z > 2$ and let*

$$I(z) = \begin{cases} z\left(\ln T_z - \ln \xi\right) - \ln\left(e^{T_z} - T_z - 1\right) + \ln\left(e^{\xi} - \xi - 1\right), & \text{if } z > 2 \\ \ln 2 - 2\ln\xi + \ln(e^{\xi} - \xi - 1), & \text{if } z = 2 \end{cases}.$$

Let $\beta \in [0,1)$ and let $\beta \le q \le 1 - 2(1-\beta)/k$. Then

$$\mathbb{P}\left(X_{q,\beta}^{(1)} > 0\right) = o(1) + \binom{m_2}{qn_2}(2^k - k - 1)^{m_2 - qn_2} \exp\left(n_2 H(\beta) - km_2 H(q) - n_2(1-\beta) I\left(\frac{k(1-q)}{1-\beta}\right) + O(n_2\delta^2 \ln(1/\delta))\right).$$

Moreover, when $q < \beta$, the above probability is 0.

With the above estimate available, a substantial part of our work is devoted to the proof of the following statement.

Lemma 4. *Let $\delta > 0$ be sufficiently small. Then the following holds with probability $1-o(1)$. For any $0.7 \le \beta \le 1 - e_k\delta/2$, and any $\beta \le q \le 1 - \frac{2(1-\beta)}{k}$ we have $X_{q,\beta}^{(1)} = 0$.*

The lower bound for q in the above lemma comes from the fact that otherwise the corresponding probability is zero (cf. Lemma 3). Moreover, the upper bound in the range of q stems from the fact that the average degree of the complement of a set with t vertices and total degree $q \cdot kM_2$ is at least two. More precisely, the total degree of the core satisfies for small $\delta > 0$

$$kM_2 \ge q \cdot kM_2 + 2(N_2 - t) \implies q \le 1 - \frac{2(1-\beta)N_2}{kM_2} \overset{(4.1)}{\le} 1 - \frac{2(1-\beta)}{k}.$$

With the above result at hand we can finally complete the proof of Theorem 5.

Proof (Theorem 5). It is sufficient to show that the core C of $\widetilde{H}_{n,p,k}$ contains with probability $1-o(1)$ no maximal 1-dense subsets, where $p = ck/\binom{n-1}{k-1}$. Note also that it is sufficient to argue about subsets of size up to, say, $(1-e_k\delta/2)N_2$ as (4.1) implies that for small δ all larger subsets have density smaller than 1.

Let $k \geq 5$. By applying Lemma 1 we obtain that $H_{n,cn,k}$ does not contain any 1-dense set with less that $0.7n$ vertices, and the same is true for $\widetilde{H}_{n,p,k}$, by Proposition 2 and Theorem 6. In particular, C does not contain such a subset, and it remains to show the claim for sets of size at least $0.7n \geq 0.7N_2$. The proof is completed by applying Lemma 4, as we can choose $\delta > 0$ as small as we please.

The case $k = 3$ requires slightly more work. Lemma 2 guarantees that C has no subset with $\leq n/2$ vertices that contains exactly as many edges as vertices, and there is no edge that contains precisely two vertices in that set. In other words, by using Proposition 3, C does not contain a maximal 1-dense set with $n/2$ vertices. However, we know that with probability $1-o(1)$

$$N_2 = (1 - e^{\xi^*} - \xi^* e^{-\xi *} \pm O(\delta))n, \text{ where } 3 = \frac{\xi^*(e^{\xi^*}-1)}{e^{\xi^*}-1-\xi^*}.$$

Numerical calculations imply that $N_2 \geq 0.63n$ for any δ that is small enough. So, C does not contain any maximal 1-dense subset with less than $n/2 \leq N_2/(2 \cdot 0.63) \leq 0.77N_2$ vertices. In this case, the proof is completed again by applying Lemma 4. Finally, the case $k = 4$ was treated in [4]. This completes the proof of Theorem 5.

References

1. Bohman, T., Frieze, A.: Avoiding a Giant Component. Random Structures & Algorithms 19, 75–85 (2001)
2. Bohman, T., Frieze, A., Krivelevich, M., Loh, P., Sudakov, B.: Ramsey Games with Giants (Submitted)
3. Bohman, T., Frieze, A., Wormald, N.: Avoiding a Giant Component in Half the Edge Set of a Random Graph. Random Structures & Algorithms 25, 432–449 (2004)
4. Bohman, T., Kim, J.H.: A Phase Transition for Avoiding a Giant Component. Random Structures & Algorithms 28(2), 195–214 (2006)
5. Cain, J.A., Sanders, P., Wormald, N.: The Random Graph Threshold for k-orientiability and a Fast Algorithm for Optimal Multiple-choice Allocation. In: Bansal, N., Pruhs, K., Stein, C. (eds.) SODA 2007, pp. 469–476. SIAM, Philadelphia (2007)
6. Cooper, C.: The Cores of Random Hypergraphs with a Given Degree Sequence. Random Structures & Algorithms 25(4), 353–375 (2004)
7. Devroye, L., Morin, P.: Cuckoo Hashing: Further Analysis. Information Processing Letters 86(4), 215–219 (2003)
8. Dietzfelbinger, M., Goerdt, A., Mitzenmacher, M., Montanari, A., Pagh, R., Rink, M.: Tight Thresholds for Cuckoo Hashing via XORSAT. In: Abramsky, S., et al. (Eds.): ICALP 2010, Part I. LNCS, vol. 6198, pp. 213–225. Springer, Heidelberg (2010)

9. Dietzfelbinger, M., Schellbach, U.: On Risks of Using Cuckoo Hashing with Simple Universal Hash Classes. In: Mathieu, C. (ed.) SODA 2009, pp. 795–804. SIAM, Philadelphia (2009)
10. Dietzfelbinger, M., Weidling, C.: Balanced Allocation and Dictionaries with Tightly Packed Constant Size Bins. Theoretical Computer Science 380(1-2), 47–68 (2007)
11. Drmota, M., Kutzelnigg, R.: A Precise Analysis of Cuckoo Hashing (submitted)
12. Erdős, P., Rényi, A.: On the Evolution of Random Graphs. Publication of the Mathematical Institute of the Hungarian Academy of Sciences 5, 17–61 (1960)
13. Fernholz, D., Ramachandran, V.: The k-orientability Thresholds for $G_{n,p}$. In: Bansal, N., Pruhs, K., Stein, C. (eds.) SODA 2007, pp. 459–468. SIAM, Philadelphia (2007)
14. Fotakis, D., Pagh, R., Sanders, P., Spirakis, P.: Space efficient hash tables with worst case constant access time. In: Habib, M., Alt, H. (eds.) STACS 2003. LNCS, vol. 2607, pp. 271–282. Springer, Heidelberg (2003)
15. Frieze, A., Melsted, P.: Maximum Matchings in Random Bipartite Graphs and the Space Utilization of Cuckoo Hashtables (2009) (manuscript), http://arxiv.org/abs/0910.5535
16. Gao, P., Wormald, N.: Load Balancing and Orientability Thresholds for Random Hypergraphs. In: Proceedings of STOC (to appear, 2010)
17. Janson, S., Łuczak, T., Ruciński, A.: Random Graphs. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York (2000)
18. Kim, J.H.: Poisson Cloning Model for Random Graphs (2006) (manuscript)
19. Mitzenmacher, M., Vadhan, S.: Why Simple Hash Functions Work: Exploiting the Entropy in a Data Stream. In: Teng, S.-H. (ed.) SODA 2008, pp. 746–755. SIAM, Philadelphia (2008)
20. Molloy, M.: Cores in Random Hypergraphs and Boolean Formulas. Random Structures & Algorithms 27(1), 124–135 (2005)
21. Pagh, R., Rodler, F.F.: Cuckoo Hashing. In: auf der Heide, M. (ed.) ESA 2001. LNCS, vol. 2161, pp. 121–133. Springer, Heidelberg (2001)
22. Spencer, J., Wormald, N.: Birth Control for Giants. Combinatorica 27(5), 587–628 (2007)

On the Inapproximability of Vertex Cover on k-Partite k-Uniform Hypergraphs*

Venkatesan Guruswami and Rishi Saket

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract. Computing a minimum vertex cover in graphs and hypergraphs is a well-studied optimizaton problem. While intractable in general, it is well known that on bipartite graphs, vertex cover is polynomial time solvable. In this work, we study the natural extension of bipartite vertex cover to hypergraphs, namely finding a small vertex cover in k-uniform k-partite hypergraphs, when the k-partition is given as input. For this problem Lovász [16] gave a $\frac{k}{2}$ factor LP rounding based approximation, and a matching $\left(\frac{k}{2} - o(1)\right)$ integrality gap instance was constructed by Aharoni *et al.* [1]. We prove the following results, which are the first strong hardness results for this problem (here $\varepsilon > 0$ is an arbitrary constant):

- NP-hardness of approximating within a factor of $\left(\frac{k}{4} - \varepsilon\right)$, and
- Unique Games-hardness of approximating within a factor of $\left(\frac{k}{2} - \varepsilon\right)$, showing optimality of Lovász's algorithm under the Unique Games conjecture.

The NP-hardness result is based on a reduction from minimum vertex cover in r-uniform hypergraphs for which NP-hardness of approximating within $r-1-\varepsilon$ was shown by Dinur *et al.* [5]. The Unique Games-hardness result is obtained by applying the recent results of Kumar *et al.* [15], with a slight modification, to the LP integrality gap due to Aharoni *et al.* [1]. The modification is to ensure that the reduction preserves the desired structural properties of the hypergraph.

1 Introduction

A k-uniform hypergraph $G = (V, E)$ consists of a set of vertices V and hyperedges E where every hyperedge is a set of exactly k vertices. The hypergraph G is said to be m-colorable if there is a coloring of the vertex set V with at most m colors such that no hyperedge in E has all its vertices of the same color. We shall be interested in the stricter condition of *strong* colorability as defined in Aharoni *et al.* [1], wherein G is said to be m-strongly-colorable if there is an m-coloring such of the vertex set V such that every hyperedge E has k distinctly colored vertices. In particular a k-strongly-colorable k-uniform hypergraph is a k-partite

* Research supported in part by a Packard Fellowship and US-Israel BSF-2008293.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 360–371, 2010.

k-uniform hypergraph, where the k-partition of the vertex set corresponds to the k color classes.

A *vertex cover* of a hypergraph $G = (V, E)$ is a subset V' of vertices such that every hyperedge in E contains at least one vertex from V'. The problem of computing the vertex cover of minimum size in a (hyper)graph has been deeply studied in combinatorics with applications in various areas of optimization and computer science. This problem is known to be NP-hard. On the other hand, for k-uniform hypergraphs the greedy algorithm of picking a maximal set of disjoint hyperedges and including all the vertices in those hyperedges gives a factor k approximation. More sophisticated algorithmic techniques only marginally improve the approximation factor to $k - o(1)$ [9].

Several inapproximability results have been shown for computing the minimum vertex cover. For general k, an $\Omega(k^{1/19})$ hardness factor was first shown by Trevisan [21], subsequently strengthened to $\Omega(k^{1-\varepsilon})$ by Holmerin [10] and to a $k - 3 - \varepsilon$ hardness factor due to Dinur, Guruswami and Khot [4]. The currently best known $k - 1 - \varepsilon$ hardness factor is due to Dinur, Guruswami, Khot and Regev [5] who build upon [4] and the seminal work of Dinur and Safra [6] who showed the best known 1.36 hardness of approximation for vertex cover in graphs ($k = 2$).

All of the above mentioned results are based on standard complexity assumptions. However, assuming Khot's Unique Games Conjecture (UGC) [12], an essentially optimal $k - \varepsilon$ hardness of approximating the minimum vertex cover on k-uniform hypergraphs was shown by Khot and Regev [14]. In more recent works the UGC has been used to relate the inapproximability of various classes of constraint satisfaction problems (CSPs) to the corresponding semi-definite programming (SDP) integrality gap [19], or the linear programming (LP) integrality gap [17] [15]. The recent work of Kumar *et al.* [15] generalizes the result of [14] and shall be of particular interest in this work.

In this work we investigate the complexity of computing the minimum vertex cover in hypergraphs that are strongly colorable and where the strong coloring is given as part of the input. Variants of this problem are studied for databases related applications such as distributed data mining [7], schema mapping discovery [8] and in optimizing finite automata [11]. The particular case of computing the minimum vertex cover in k-uniform k-partite (with the partition given) hypergraphs was studied by Lovász [16] who obtained a $k/2$ approximation for it by rounding its natural LP relaxation. Subsequently, Aharoni, Holzman and Krivelevich [1] proved a tight integrality gap of $k/2 - o(1)$ for the LP relaxation. On the hardness side, [11] and [8] give reductions from 3SAT to it, which imply that the problem is APX-hard. However, to the best of our knowledge no better hardness of approximation was known for this problem.

In this work we show a $\left(\frac{k}{4} - \varepsilon\right)$ hardness of approximation factor for computing the minimum vertex cover on k-uniform k-partite hypergraphs. Actually, we prove a more general hardness of approximation factor of $\frac{(m-(k-1))(k-1)}{m} - \varepsilon$ for computing the minimum vertex cover in m-strongly colorable k-uniform hypergraphs. The result for k-uniform k-partite hypergraphs follows by a simple

reduction. Our results are based on a reduction from minimum vertex cover in k-uniform hypergraphs for which, as mentioned above, the best known factor $k-1-\varepsilon$ hardness of approximation factor was given in [5].

We also study the results of [15] in the context of the problems we consider. In [15], the authors proved that LP integrality gaps for a large class of monotone constraint satisfaction problems, such as vertex cover, can be converted into corresponding UGC based hardness of approximation results. As presented, the reduction in [15] does not guarantee that the structural properties of the integrality gap will be carried through into the final instance. Nevertheless, we observe that the integrality gap instance of [1] can be combined with the work of [15] with only a slight modification to yield an essentially optimal $k/2-o(1)$ factor hardness of approximation for computing the minimum vertex cover in k-uniform k-partite hypergraphs, i.e. the final instance is also guaranteed to be a k-uniform k-partite hypergraph. Similar tight inapproximability can also be obtained for a larger class of hypergraphs which we shall define later.

Main Results. We summarize the main results of this paper in the following informal statement.

Theorem *(Informal). For every $\varepsilon > 0$, and integers $k \geqslant 3$ and $m \geqslant 2k$, it is NP-hard to approximate the minimum vertex cover on m-strongly-colorable k-uniform hypergraphs to within a factor of*

$$\frac{(m-(k-1))(k-1)}{m} - \varepsilon.$$

In addition, it is NP-hard to approximate the minimum vertex cover on k-uniform k-partite hypergraphs to within a factor of $\frac{k}{4}-\varepsilon$, and within a factor of $\frac{k}{2}-\varepsilon$ assuming the Unique Games conjecture.

We now proceed to formally defining the problems we consider, followed by a discussion of the previous work and a precise statement of our results on these problems.

2 Problem Definitions

We now define the variants of the hypergraph vertex cover problem studied in this paper.

Definition 1. *For any integer $k \geqslant 2$, an instance $G=(V,E)$ of the hypergraph vertex cover problem* HypVC(k)*, is a k-uniform hypergraph (possibly weighted) where the goal is to compute a vertex cover $V' \subseteq V$ of minimum weight.*

Definition 2. *For any integers $m \geqslant k \geqslant 2$, an instance of $G=(V,E)$ of* StrongColored-HypVC(m,k) *is a m-strongly-colorable k-uniform hypergraph where the m-strong-coloring of G is given. Formally, a partition of V into m disjoint subsets (color classes) V_i $(1 \leqslant i \leqslant m)$ is given, such that every hyperedge in E has at most one vertex from each color class. In other words, every hyperedge contains k distinctly colored vertices. The goal is to compute the minimum weight vertex cover in G.*

Definition 3. *For any integer $k \geqslant 2$, an instance $G = (V, E)$ of* HypVCPartite(k) *is an k-uniform k-partite hypergraph with the k-partition given as input. The goal is to compute the minimum weight vertex cover in G. Note that* HypVCPartite(k) *is the same as* StrongColored-HypVC(k, k).

The following definition generalizes the class of k-partite hypergraphs and defines the minimum vertex cover problem for that class.

Definition 4. *For any integer $k \geqslant 2$ and positive integers $p_1, \ldots, p_k$, a hypergraph $G = (V, E)$ is called $(p_1, \ldots, p_k)$-split if there is a k-partitioning $V_1, \ldots, V_k$ of the vertex set V such that for every hyperedge $e \in E$, $|e \cap V_i| = p_i$ for $1 \leqslant i \leqslant k$.* HypVCSplit$(r, k, p_1, \ldots, p_i)$ *denotes the problem of computing the minimum vertex cover in $(p_1, \ldots, p_k)$-split r-uniform hypergraphs where the k-partitioning is given as input. Here $\sum_{i=1}^{k} p_i = r$. Note that* HypVCPartite(k) *is the same as* HypVCSplit$(k, k, 1, \ldots, 1)$.

3 Previous Work and Our Results

3.1 Previous Results

Let LP_0 be the natural "covering" linear programming relaxation for hypergraph vertex cover (see, for example, Section 1 of [1]). The linear program is oblivious to the structure of the hypergraph and can be applied to any of the variants of hypergraph vertex cover defined above. The following theorem, first proved by Lovász [16] gives an upper bound on the integrality gap of the relaxation LP_0 for HypVCPartite(k). All the upper bounds on the integrality gap stated in this section are achieved using polynomial time rounding procedures for LP_0.

Theorem 1. *(Lovász [16]) For any integer $k \geqslant 2$, for any instance G of* HypVCPartite(k),

$$\frac{\mathrm{OPT}_{\mathrm{VC}}(G)}{\mathrm{VAL}_{\mathrm{LP}_0}(G)} \leqslant \frac{k}{2} \tag{1}$$

where $\mathrm{OPT}_{\mathrm{VC}}(G)$ is the weight of the minimum vertex cover in G and $\mathrm{VAL}_{\mathrm{LP}_0}(G)$ is the optimum value of the objective function of the relaxation LP_0 applied to G.

We observe that the relaxation LP_0 does not utilize the k-partiteness property of the input hypergraph. Therefore, the upper bound in Equation (1) holds irrespective of whether the k-partition is given as input. On the other hand, the k-partition is necessary for the efficient rounding algorithm given by the previous theorem. We note that for general k-uniform hypergraphs the gap between the size of the minimum vertex cover and value of the LP solution can be as high as $k - o(1)$. The following theorem states that Equation (1) is essentially tight.

Theorem 2. *(Aharoni* et al. *[1]) The integrality gap of LP_0 on instances of* HypVCPartite(k) *is $k/2 - o(1)$.*

For instances of StrongColored-HypVC(m, k) Aharoni *et al.* [1] proved lower and upper bounds summarized in the following theorem.

Theorem 3. *(Aharoni* et al. *[1]) Let G be an instance of* StrongColored-HypVC(m,k) *where $m,k \geqslant 2$. If $m \geqslant k(k-1)$ then,*

$$\frac{\mathrm{OPT}_{\mathrm{VC}}(G)}{\mathrm{VAL}_{\mathrm{LP}_0}(G)} \leqslant \frac{(m-k+1)k}{m} \tag{2}$$

In addition, the integrality gap of LP_0 *when $m \geqslant k(k-1)$ is $\frac{(m-k+1)k}{m} - o(1)$. On the other hand, when $k < m < k(k-1)$ then,*

$$\frac{\mathrm{OPT}_{\mathrm{VC}}(G)}{\mathrm{VAL}_{\mathrm{LP}_0}(G)} \leqslant \frac{mk}{m+k} + \min\left\{\frac{m-k}{2m}a, \frac{k}{m}(1-a)\right\}, \tag{3}$$

where $a = \frac{m^2}{m+r} - \left\lfloor \frac{m^2}{m+r} \right\rfloor$.

Theorems 1 and 2 were generalized by [1] to split hypergraphs as defined in Definition 4. Their general result is stated below.

Theorem 4. *(Aharoni* et al. *[1]) For any positive integers $r, k, p_1, \ldots, p_k$ such that $\sum_{i=1}^{k} p_i = r \geqslant 2$, and any instance G of* HypVCSplit$(r, k, p_1, \ldots, p_k)$*,*

$$\frac{\mathrm{OPT}_{\mathrm{VC}}(G)}{\mathrm{VAL}_{\mathrm{LP}_0}(G)} \leqslant \max\left\{\frac{r}{2}, p_1, \ldots, p_k\right\}. \tag{4}$$

In addition, the integrality gap of LP_0 *on instances of* HypVCSplit$(r, k, p_1, \ldots, p_k)$ *is* $\max\left\{\frac{r}{2}, p_1, \ldots, p_k\right\} - o(1)$.

The following theorem states the best known NP-hardness of approximation for the minimum vertex on general hypergraphs.

Theorem 5. *(Dinur* et al. *[5]) For any $\varepsilon > 0$ and integer $k \geqslant 3$, it is NP-hard to approximate* HypVC(k) *to a factor of $k - 1 - \varepsilon$.*

The above hardness of approximation for general k is not known to be tight. On the other hand, assuming the Unique Games Conjecture one can obtain optimal inapproximability factors of $k - o(1)$ for HypVC(k). The following formal statement was proved by Khot and Regev [14].

Theorem 6. *(Khot* et al. *[14]) Assuming the Unique Games Conjecture of Khot [12], For any $\varepsilon > 0$, it is NP-hard to approximate* HypVC(k) *to within a factor of $k - \varepsilon$.*

Remark 1. A recent paper by Bansal and Khot [3] shows a strong hardness result assuming the UGC for distinguishing between a k-uniform hypergraph that is *almost* k-partite and one which has no vertex cover containing at most a $(1-\varepsilon)$ fraction of vertices (for any desired $\varepsilon > 0$). We note that this is very different from our problem where the input is always k-partite with a given k-partition (and in particular has an easily found vertex cover with a $1/k$ fraction of vertices, namely the smallest of the k parts).

3.2 Our Results

NP-hardness results. We prove the following theorem on the NP-hardness of approximating the minimum vertex cover on strongly colorable hypergraphs.

Theorem 7. *For every $\varepsilon > 0$ and integer $m \geqslant k \geqslant 3$ (such that $m \geqslant 2k$), it is NP hard to approximate* StrongColored-HypVC(m,k) *to within a factor of*

$$\frac{(m-(k-1))(k-1)}{m} - \varepsilon.$$

The above theorem is proved in Section 4 via a reduction from HypVC(k) to StrongColored-HypVC(m,k). A simple reduction from StrongColored-HypVC(k,k') also shows the following hardness results for HypVCpartite(k) and HypVCsplit$(r,k,p_1,\ldots,p_k)$. We prove Theorem 8 in Section 5 while we omit the proof of Theorem 9 due to lack of space.

Theorem 8. *For every $\varepsilon > 0$ and integer $k > 16$, it is NP-hard to approximate* HypVCpartite(k) *within a factor of $\frac{k}{4} - \varepsilon$.*

Theorem 9. *For every $\varepsilon > 0$, and positive integers $r, k, p_1, \ldots, p_k$ such that $\sum_{i=1}^{k} p_i = r \geqslant 3$ and $t := \max\{p_1,\ldots,p_k\} \geqslant 3$, it is NP-hard to approximate* HypVCsplit$(r,k,p_1,\ldots,p_k)$ *to within a factor of*

$$\max\left\{\frac{r}{4}, t-1\right\} - \varepsilon.$$

The above hardness of approximation results do not quite match the algorithmic results in Theorem 4. The next few paragraphs illustrate how recent results of [15] can be combined with the integrality gaps given in Theorems 1 and 4 to yield tight inapproximability for the corresponding problems.

Unique Games hardness. In recent work Kumar, Manokaran, Tulsiani and Vishnoi [15] have shown that for a large class of monotone constraint problems, including hypergraph vertex cover, integrality gaps for a natural LP relaxation can be transformed into corresponding hardness of approximation results based on the Unique Games Conjecture.

The reduction in [15] is analyzed using the general bounds on noise correlation of functions proved by Mossel [18]. For this purpose, the reduction perturbs a "good" solution, say x^*, to the LP relaxation for the integrality gap $G_{\mathcal{I}} = (V_{\mathcal{I}}, E_{\mathcal{I}})$, so that x^* satisfies the property that all variables are integer multiples of some $\varepsilon > 0$. Therefore, the number of distinct values in x^* is $m \approx 1/\varepsilon$. The reduction is based on a "dictatorship test" over the set $[m] \times \{0,1\}^r$ (for some parameter r) and the hardness of approximation obtained is related to the performance of a certain (efficient) rounding algorithm on x^*, which returns a solution no smaller than the optimum on $G_{\mathcal{I}}$. As described in [15] the reduction is not guaranteed to preserve structural properties of the integrality gap instance $G_{\mathcal{I}}$, such as strong colorability or k-partiteness.

We make the simple observation that the dictatorship test in the above reduction can analogously be defined over $V_{\mathcal{I}} \times \{0,1\}^r$ which then preserves strong

colorability and partiteness properties of $G_{\mathcal{I}}$ into the final instance. The gap obtained depends directly on the optimum in $G_{\mathcal{I}}$. This observation, combined with the result of [15] and the integrality gap for HypVCPartite(k) stated in Theorem 1 yields the following optimal UGC based hardness result.

Theorem 10. *Assuming the Unique Games Conjecture, it is NP-hard to approximate* HypVCPartite(k) *to within a factor of* $\frac{k}{2} - \varepsilon$ *for any* $\varepsilon > 0$.

Due to lack of space we omit the proof and refer the reader to the full version of this paper and [15] for details of the analysis. Similar inapproximability results can be obtained for StrongColored-HypVC(m,k) and HypVCsplit$(r,k,p_1,\ldots,p_k)$ using the corresponding integrality gaps given in Theorems 3 and 4.

4 Reduction from HypVC(k) to StrongColored-HypVC(m,k) and Proof of Theorem 7

Let k and m be two positive integers such that $m \geqslant k \geqslant 2$. In this section we give a reduction from an instance of HypVC(k) to an instance of StrongColored-HypVC(m,k).

Reduction. Let the $H = (U,F)$ be an instance of HypVC(k), i.e. H is a k-uniform hypergraph with vertex set U, and a set F of hyperedges. The reduction constructs an instance $G = (V,E)$ of StrongColored-HypVC(m,k) where G is an k-uniform, m-strongly colorable hypergraph, i.e. $V = \cup_{i=1}^{m} V_i$, where V_i are m disjoint subsets (color classes) such that every hyperedge in E has exactly one vertex from each subset. The main idea of the reduction is to let new vertex set V be the union of m copies of U, and for every hyperedge $e' \in F$, add all hyperedges which contain exactly one copy (in V) of every vertex in e', and at most one vertex from any of the m copies of U (in V). Clearly every hyperedge 'hits' any of the m copies of U in V at most once which naturally gives an m-strong coloring of V. It also ensures that if there is a vertex cover in G which is the union of a subset of the copies of U, then it must contain at least $m-k+1$ of the copies. Our analysis shall essentially build upon this idea.

To formalize the reduction we first need to define a useful notation.

Definition 5. *Given a hyperedge* $e' = \{u_1,\ldots,u_k\}$ *in* F*, and a subset* $I \subseteq [m]$ *where* $|I| = k$*, a mapping* $\sigma : I \mapsto \{u_1,\ldots,u_k\}$ *is said to be a "*(I,e')*-matching" if* σ *is a one-to-one map. Let* $\Gamma_{I,e'}$ *be the set of all* (I,e')*-matchings. Clearly,* $|\Gamma_{I,e'}| = k!$ *, for all* $I \subseteq [m], |I| = k$ *and* $e' \in F$.

The steps of the reduction are as follows.

1. For $i = 1,\ldots,m$, let $V_i = U \times \{i\}$
2. For every hyperedge e' in F, for every subset $I \subseteq [m]$ such that $|I| = k$, for every (I,e')-matching $\sigma \in \Gamma_{I,e'}$ we add the hyperedge $e = e(e',I,\sigma)$ which

is defined as follows.

$$\forall i \in [m], \quad V_i \cap e = \begin{cases} (\sigma(i), i) & \text{if } i \in I \\ \emptyset & \text{otherwise.} \end{cases} \tag{5}$$

The above reduction outputs the instance $G = (V, E)$ of StrongColored-HypVC(m, k). Note that the vertex set V is of size $m|U|$ and for every hyperedge $e' \in F$ the number of hyperedges added in E is $\binom{m}{k} \cdot k!$. Therefore the reduction is polynomial time. In the next section we present the analysis of this reduction.

Analyzing the Reduction

Theorem 11. *Let C be the size of the optimal vertex cover in $H = (U, F)$, and let C' be the size of the optimal vertex cover in $G = (V, E)$. Then,*

$$(m - (k-1))C \leqslant C' \leqslant mC$$

Using the above theorem we can complete the proof of Theorem 7 as follows.

Proof. (of Theorem 7) Theorem 11 combined with the $k-1-\varepsilon$ inapproximability for HypVC(k) given by [5] and stated in Theorem 5, implies an inapproximability of,

$$\frac{(m - (k-1))C_1}{mC_2},$$

for some integers C_1, C_2 (depending on H) such that $C_1 \geqslant C_2(k - 1 - \varepsilon)$ for some $\varepsilon > 0$. It is easy to see that the above expression can be simplified to yield

$$\frac{(m - (k-1))(k-1)}{m} - \varepsilon' \tag{6}$$

as the inapproximability factor for StrongColored-HypVC(m, k). This proves Theorem 7. □

Proof. (of Theorem 11) We first show that there is a vertex cover of size at most mC in G, where C is the size of an optimal vertex cover U^* in H. To see this consider the set $V^* \subseteq V$, where $V^* = U^* \times [m]$. For every hyperedge $e' \in F$, $e' \cap U^* \neq \emptyset$, and therefore $e \cap U^* \times \{i\} \neq \emptyset$, for some $i \in [m]$, for all $e = e(e', I, \sigma)$. Therefore, $V^* \cap e \neq \emptyset$ for all $e \in E$. The size of V^* is mC which proves the upper bound in Theorem 11. In the rest of the proof we shall prove the lower bound in Theorem 11.

Let S be the optimal vertex cover in G. Our analysis shall prove a lower bound on the size S in terms of the size of the optimal vertex cover in H. Let $S_i := V_i \cap S$ for $i \in [m]$. Before proceeding we introduce the following useful quantity. For every $Y \subseteq [m]$, we let $A_Y \subseteq U$ be the set of all vertices which have a copy in S_i for some $i \in Y$. Formally,

$$A_Y := \{u \in U \mid \exists i \in Y \text{ s.t. } (u, i) \in S_i\}.$$

The following simple lemma follows from the construction of the edges E in G.

Lemma 1. *Let $I \subseteq [m]$ be any subset such that $|I| = k$. Then A_I is a vertex cover of the hypergraph H.*

Proof. Fix any subset I as in the statement of the lemma. Let $e' \in F$ be any hyperedge in H. For a contradiction assume that $A_I \cap e' = \emptyset$. This implies that the sets S_i $(i \in I)$ do not have a copy of any vertex in e'. Now choose any $\sigma \in \Gamma_{I,e'}$ and consider the edge $e(e', I, \sigma) \in E$. This edge can be covered only by vertices in V_i for $i \in I$. However, since S_i does not contain a copy of any vertex in e' for $i \in I$ the edge $e(e', I, \sigma)$ is not covered by S which is a contradiction. This completes the proof. □

The next lemma combines the previous lemma with the minimality of S to show a strong structural statement for S, that any S_i is "contained" in the union of any other k sets S_j. It shall enable us to prove that most of the sets S_i are large.

Lemma 2. *Let $I \subseteq [m]$ be any set of indices such that $|I| = k$. Then, for any $j' \in [m]$, $S_{j'} \subseteq A_I \times \{j'\}$.*

Proof. Let I be any choice of a set of k indices in $[m]$ as in the statement of the lemma. From Lemma 1 we know that A_I is a vertex cover in H and is therefore non-empty. Let $j' \in [m]$ be an arbitrary index for which we shall verify the lemma for the above choice of I. If $j' \in I$, then the lemma is trivially true. Therefore, we may assume that $j' \notin I$. For a contradiction we assume that,

$$(u, j') \in S_{j'} \setminus (A_I \times \{j'\}) \tag{7}$$

From the minimality of S, we deduce that there must be a hyperedge, say $e \in E$ such that e is covered by (u, j') and by no other vertex in S; otherwise $S \setminus \{(u, j')\}$ would be a smaller vertex cover in G. Let $e = e(e', I', \sigma)$ for some $e' \in F$, $I' \subseteq [m]$ $(|I'| = k)$ and $\sigma \in \Gamma_{I',e'}$. Now, since (u, j') covers e, we obtain that $j' \in I'$ and $\sigma(j') = u \in e'$. Combining this with the fact that $j' \notin I$, and that $|I| = |I'| = k$, we obtain that $I \setminus I' \neq \emptyset$.

Let $j \in I \setminus I'$. We claim that $(u, j) \notin S_j$. To see this, observe that if $(u, j) \in S_j$ then $u \in A_I$ which would contradict our assumption in Equation (7).

We now consider the following hyperedge $\tilde{e} = \tilde{e}(e', \tilde{I}, \tilde{\sigma}) \in E$ where the quantities are defined as follows. The set $\tilde{I}$ simply replaces the index j' in I' with the index j, i.e.

$$\tilde{I} = (I' \setminus \{j'\}) \cup \{j\}. \tag{8}$$

Analogously, $\tilde{\sigma} \in \Gamma_{\tilde{I},e'}$ is identical to σ except that it is defined on j instead of j' where $\tilde{\sigma}(j) = \sigma(j') = u$. Formally,

$$\tilde{\sigma}(i) = \begin{cases} \sigma(i) & \text{if } i \in \tilde{I} \setminus \{j\} \\ u & \text{if } i = j \end{cases} \tag{9}$$

Equations (8) and (9) imply the following,

$$V_i \cap \tilde{e} = V_i \cap e \; \forall i \in [m] \setminus \{j, j'\} \tag{10}$$

$$V_j \cap \tilde{e} = (u, j) \tag{11}$$

$$V_{j'} \cap \tilde{e} = \emptyset \tag{12}$$

Since $(u, j') \in S$ uniquely covers e, Equation (10) implies that $\tilde{e}$ is not covered by any vertex in S_i for all $i \in [m] \setminus \{j, j'\}$. Moreover, since $j' \notin \tilde{I}$ no vertex in $S_{j'}$ covers $\tilde{e}$. On the other hand, by our assumption in Equation (7) $(u, j) \notin S_j$, which along with Equation (11) implies that no vertex in S_j covers $\tilde{e}$. Therefore, $\tilde{e}$ is not covered by S. This is a contradiction to the fact that S is a vertex cover in G and therefore our assumption in Equation (7) is incorrect. This implies that $S_{j'} \subseteq A_I \times \{j'\}$. This holds for every j', thus proving the lemma. □

Note that the above lemma immediately implies the following corollary.

Corollary 1. *For every $I \subseteq [m], |I| = k$, we have $A_{[m]} = A_I$.*

It is easy to see the following simple lemma.

Lemma 3. *For any vertex $u \in A_{[m]}$, let $I_u \subseteq [m]$ be the largest set such that $u \notin A_{I_u}$. Then, $|I_u| < k$.*

Proof. Suppose the above does not hold. Then I_u (or any subset of I_u of size k) would violate Corollary 1, which is a contradiction. This completes the proof. □

The above lemma immediately implies the desired lower bound on the size of S.

Lemma 4. *Let C be the size of the optimal vertex cover in H. Then,*

$$|S| \geqslant (m - (k-1))C.$$

Proof. For convenience, let $q = |A_{[m]}|$. Note that, by Lemma 1 $A_{[m]}$ is a vertex cover in H. Therefore, $q \geqslant C$. From Lemma 3 we deduce that every vertex $u \in A_{[m]}$ has a copy (u, i) in at least $m - (k-1)$ of the sets S_i. Therefore, S contains $m - (k-1)$ copies of every vertex in $A_{[m]}$ which yields,

$$|S| \geqslant (m - (k-1))q \geqslant (m - (k-1))C,$$

thus completing the proof. □

The above also completes the proof of the lower bound of Theorem 11. □

5 Reduction from StrongColored-HypVC(k, k') to HypVCpartite(k) and Proof of Theorem 8

We prove Theorem 8 by giving a simple reduction from an instance $G = (V, E)$ of StrongColored-HypVC(k, k') to an instance $G' = (V', E')$ of HypVC-partite(k) where the parameters will be chosen later.

Given $G = (V, E)$ construct V' by adding k "dummy" vertices $b_1, \ldots, b_k$ to V, i.e. $V' = V \cup \{b_1, \ldots, b_k\}$. Let $V_1, \ldots, V_k$ be the k color classes of V. For any hyperedge $e \in E$, construct a corresponding hyperedge $e' \in E'$ which contains all the vertices in e in addition to b_i if $e \cap V_i = \emptyset$ for all $i \in [k]$. It is easy to see that G' is a k-partite hypergraph with the k-partition given by the subsets

$V_i \cup \{b_i\}$. As a final step, set the weight of the dummy vertices $b_1, \ldots, b_k$ to be much larger than $|V'|$ so that no dummy vertex is chosen in any optimal vertex cover in G'. This is because V is always a vertex cover in G. Note that the hypergraph can be made unweighted by the (standard) technique of replicating each dummy vertex many times and multiplying the hyperedges appropriately.

Since no optimal vertex cover in G' contains a dummy vertex we deduce that an optimal vertex cover in G' is an optimal vertex cover in G and vice versa. From Theorem 7, for any $\varepsilon > 0$, we obtain a hardness factor of,

$$\frac{(k-(k'-1))(k'-1)}{k} - \varepsilon,$$

for approximating HypVCpartite(k). Let $\alpha := \frac{(k'-1)}{k}$. The above expression is maximized in terms of k when $\frac{(1-\alpha)\alpha}{2}$ attains a maximum where $\alpha \in [0,1]$. Clearly, the maximum is obtained when $\alpha = \frac{(k'-1)}{k} = \frac{1}{2}$, thus yielding as the hardness of approximation factor:

$$\left(\frac{k'-1}{2}\right) - \varepsilon = \left(\frac{k}{4}\right) - \varepsilon,$$

which proves Theorem 8.

Acknowledgment. The authors would like to thank Anupam Gupta for suggestions which helped simplify the analysis.

References

1. Aharoni, R., Holzman, R., Krivelevich, M.: On a theorem of Lovász on covers in r-partite hypergraphs. Combinatorica 16(2), 149–174 (1996)
2. Austrin, P., Mossel, E.: Approximation resistant predicates from pairwise independence. In: IEEE Conference on Computational Complexity, pp. 249–258 (2008)
3. Bansal, N., Khot, S.: Inapproximability of hypergraph vertex cover and applications to scheduling problems. In: Proceedings of ICALP (July 2009)
4. Dinur, I., Guruswami, V., Khot, S.: Vertex cover on k-uniform hypergraphs is hard to approximate within factor (k-3-ε). ECCC Technical Report TR02-027 (2002)
5. Dinur, I., Guruswami, V., Khot, S., Regev, O.: A new multilayered PCP and the hardness of hypergraph vertex cover. In: Proc. 35^{th} ACM STOC, pp. 595–601 (2003)
6. Dinur, I., Safra, S.: The importance of being biased. In: Proc. 34^{th} ACM STOC, pp. 33–42 (2002)
7. Feder, T., Motwani, R., O'Callaghan, L., Panigrahy, R., Thomas, D.: Online distributed predicate evaluation. Technical Report, Stanford University (2003)
8. Gottlob, G., Senellart, P.: Schema mapping discovery from data instances. J. ACM 57(2) (January 2010)
9. Halperin, E.: Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. SIAM J. Comput. 31(5), 1608–1623 (2002)

10. Holmerin, J.: Improved inapproximability results for vertex cover on k -uniform hypergraphs. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 1005–1016. Springer, Heidelberg (2002)
11. Ilie, L., Solis-Oba, R., Yu, S.: Reducing the size of NFAs by using equivalences and preorders. In: Apostolico, A., Crochemore, M., Park, K. (eds.) CPM 2005. LNCS, vol. 3537, pp. 310–321. Springer, Heidelberg (2005)
12. Khot, S.: On the power of unique 2-prover 1-round games. In: Proc. 34^{th} ACM STOC, pp. 767–775 (2002)
13. Khot, S., Kindler, G., Mossel, E., O'Donnell, R.: Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?. SIAM J. Comput. 37(1), 319–357 (2007)
14. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within 2-epsilon. J. Comput. Syst. Sci. 74(3), 335–349 (2008)
15. Kumar, A., Manokaran, R., Tulsiani, M., Vishnoi, N.K.: On the optimality of a class of LP-based algorithms (2009) (manuscript)
16. Lovász, L.: On minimax theorems of combinatorics. Doctoral Thesis, Mathematiki Lapok 26, 209–264 (1975)
17. Manokaran, R., Naor, J., Raghavendra, P., Schwartz, R.: SDP gaps and UGC hardness for multiway cut, 0-extension, and metric labeling. In: Proc. 40^{th} ACM STOC, pp. 11–20 (2008)
18. Mossel, E.: Gaussian bounds for noise correlation of functions and tight analysis of long codes. In: Proc. 49^{th} IEEE FOCS, pp. 156–165 (2008)
19. Raghavendra, P.: Optimal algorithms and inapproximability results for every CSP? In: Proc. 40^{th} ACM STOC, pp. 245–254 (2008)
20. Samorodnitsky, A., Trevisan, L.: Gowers uniformity, influence of variables, and PCPs. SIAM J. Comput. 39(1), 323–360 (2009)
21. Trevisan, L.: Non-approximability results for optimization problems on bounded degree instances. In: Proc. 33^{rd} ACM STOC, pp. 453–461 (2001)

Dynamic Programming for Graphs on Surfaces*

Juanjo Rué[1], Ignasi Sau[2], and Dimitrios M. Thilikos[3]

[1] Laboratorie d'Informatique, École Polytechnique, 91128 Palaiseau-Cedex, France
rue1982@lix.polytechnique.fr
[2] Department of Computer Science, Technion, Haifa, Israel
ignasi@cs.technion.ac.il
[3] Dept. of Mathematics, National and Kapodistrian University of Athens, Greece
sedthilk@math.uoa.gr

Abstract. We provide a framework for the design and analysis of dynamic programming algorithms for surface-embedded graphs on n vertices and branchwidth at most k. Our technique applies to general families of problems where standard dynamic programming runs in $2^{\mathcal{O}(k \cdot \log k)} \cdot n$ steps. Our approach combines tools from topological graph theory and analytic combinatorics. In particular, we introduce a new type of branch decomposition called *surface cut decomposition*, capturing how partial solutions can be arranged on a surface. Then we use *singularity analysis* over expressions obtained by the *symbolic method* to prove that partial solutions can be represented by a single-exponential (in the branchwidth k) number of configurations. This proves that, when applied on surface cut decompositions, dynamic programming runs in $2^{\mathcal{O}(k)} \cdot n$ steps. That way, we considerably extend the class of problems that can be solved in running times with a *single-exponential dependence* on branchwidth and unify/improve all previous results in this direction.

Keywords: analysis of algorithms; parameterized algorithms; analytic combinatorics; graphs on surfaces; branchwidth; dynamic programming; polyhedral embeddings; symbolic method; non-crossing partitions.

1 Introduction

One of the most important parameters in the design and analysis of graph algorithms is the branchwidth of a graph. Branchwidth, together with its twin parameter of treewidth, can be seen as a measure of the topological resemblance of a graph to a tree. Its algorithmic importance dates back in the celebrated theorem of Courcelle (see e.g. [6]), stating that graph problems expressible in Monadic Second Order Logic can be solved in $f(\mathbf{bw}) \cdot n$ steps (here **bw** is the

* Research supported by the European Research Council under the EC's 7th Framework Programme, ERC grant agreement 208471 - ExploreMaps project, the Israel Science Foundation, grant No. 1249/08, and the project "Kapodistrias" (ΑΠ 02839/28.07.2008) of the National and Kapodistrian University of Athens.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 372–383, 2010.

branchwidth[1] and n is the number of vertices of the input graph). Using parameterized complexity terminology, this implies that a large number of graph problems are fixed-parameter tractable when parameterized by the branchwidth of their input graph. As the bounds for $f(\mathbf{bw})$ provided by Courcelle's theorem are huge, the design of tailor-made dynamic programming algorithms for specific problems so that $f(\mathbf{bw})$ is a simple –preferably a *single-exponential*– function, became a natural (and unavoidable) ingredient for many results on graph algorithms (see [3,4,20,10]). In this paper, we provide a general framework for the design and analysis of dynamic programming algorithms for graphs embedded in surfaces where $f(\mathbf{bw}) = 2^{\mathcal{O}(\mathbf{bw})}$.

Dynamic programming. Dynamic programming is applied in a bottom-up fashion on a rooted branch decomposition of the input graph G, that roughly is a way to decompose the graph into a tree structure of edge bipartitions (the formal definition is in Section 2). Each bipartition defines a separator S of the graph called *middle set*, of cardinality bounded by the branchwidth of the input graph. The decomposition is routed in the sense that one of the parts of each bipartition is the "lower part of the middle set", i.e., the so-far processed one. For each graph problem, dynamic programming requires the suitable definition of tables encoding how potential (global) solutions of the problem are restricted to a middle set and the corresponding lower part. The size of these tables reflects the dependence on $k = |S|$ in the running time of the dynamic programming.

Designing the tables for each middle set S is not always an easy task and may vary considerably due to the particularities of each problem. The simplest cases are problems such as Vertex Cover and Dominating Set, where the certificate of the solution is a set of vertices whose choice is not restricted by some global condition. This directly yields to the desired $2^{\mathcal{O}(k)}$ upper bound on their size. For other problems, such as Longest Path, Cycle Packing, or Hamiltonian Cycle, things are more complicated as the tables encode *pairings of vertices of* S, which are $2^{\Theta(k \log k)}$ many. However, for such problems one can do better for *planar graphs* following the approach introduced in [12]. The idea in [12] is to use a special type of branch decomposition called *sphere cut decomposition* (introduced in [19]) that can guarantee that the pairings are *non-crossing* pairings around a virtual edge-avoiding cycle (called *noose*) of the plane where G is embedded. This restricts the number of tables corresponding to a middle set S by the k-th Catalan number, which is *single-exponential* in k. The same approach was extended for graphs embedded in surfaces of genus γ [9]. The idea in [9] was to perform a *planarization* of the input graph by splitting the potential solution into at most γ pieces and then applying the sphere cut decomposition technique of [12] to a more general version of the problem where the number of pairings is still bounded by some Catalan number (see also [11] for the application of this technique for more general graphs).

[1] The original statement of Courcelle's theorem used the parameter of treewidth instead of branchwidth. The two parameters are approximately equivalent, in the sense that one is a constant-factor approximation of the other.

A wider family of problems are those where the tables of dynamic programming encode *packings* of S into sets; throughout this paper, we call these problems *packing-encodable problems*. Typical problems of this type are Connected Vertex Cover, Feedback Vertex Set, and Steiner Tree, where the connected components of a potential solution can be encoded by a collection of disjoint subsets of S, each of of *arbitrary cardinality*. Here, the general bound on the table size is given by the k-th Bell number, and thus again by $2^{\Theta(k \cdot \log k)}$. (To exemplify the differences between distinct encodings, typical dynamic programming algorithms for Vertex Cover and Connected Vertex Cover can be found in [18].) Unfortunately, for the latter category of problems, none of the current techniques is able to drop this bound to a single-exponential one for graphs embedded in surfaces.

Our results. In this paper, we follow a different approach in order to design single-exponential (in **bw**) algorithms for graphs embedded in surfaces. In particular, we deviate significantly from the planarization technique of [9], which is not able to tackle problems whose solutions are encoded by general packings. Instead, we extend the concept of sphere cut decomposition from planar graphs to graphs embeddable in generic surfaces, and we exploit directly the combinatorial structure of the potential solutions in the topological surface. Our approach permits us to provide in a unified way a single-exponential (in **bw**) time analysis for all aforementioned problems. Examples of other such problems are Connected Dominating Set, Connected r-Domination, Connected FVS, Maximum Leaf Spanning Tree, Maximum Full-Degree Spanning Tree, Maximum Eulerian Subgraph, or Maximum Leaf Tree. Our results imply all the results in [9,12], and with running times whose genus dependence is better than the ones in [9], as discussed in Section 6.

Our techniques. For our results we enhance the current technology of dynamic programming with new tools for both topological graph theory and analytic combinatorics. We first propose a special type of branch decomposition of embedded graphs with nice topological properties, which we call *surface cut decomposition* (see Section 4). Roughly, the middle sets of such a decomposition are situated along a bounded (by the genus γ) set of nooses of the surface with few (again bounded by γ) common points. Such a decomposition is based on the concept of *polyhedral decomposition* introduced in Section 3. We prove that the sizes of the tables of the dynamic programming correspond to the number of non-crossing partitions of vertex sets lying in the boundary of a generic surface. To count these partitions, we use a powerful technique of analytic combinatorics: *singularity analysis* over expressions obtained by the *symbolic method* (for more on this technique, see the monograph of Flajolet and Sedgewick [13]). The symbolic method gives a precise asymptotic enumeration of the number of non-crossing partitions, that yields the single-exponentiality of the table size (see Section 5). As this is the first time such a counting is done, our combinatorial results have independent mathematical interest.

For performing dynamic programming, our approach resides in a common preprocessing step that constructs a *surface cut decomposition* (Algorithm 2

in Section 4). Then, what remains is just to run the dynamic programming algorithm on such a surface cut decomposition. The exponential bound on the size of the tables of this dynamic programming algorithm is provided as a result of our analysis (Theorem 4 of Section 6). Due to space limitations, this extended abstract contains no proofs; they can be found in [18].

2 Preliminaries

Topological surfaces. In this paper, surfaces are compact and their boundary is homeomorphic to a finite set (possibly empty) of disjoint circles. We denote by $\beta(\Sigma)$ the number of connected components of the boundary of a surface Σ. The Surface Classification Theorem [16] asserts that a compact and connected surface without boundary is determined, up to homeomorphism, by its Euler characteristic $\chi(\Sigma)$ and by whether it is orientable or not. More precisely, orientable surfaces are obtained by adding $g \geq 0$ *handles* to the sphere $\mathbb{S}^2$, obtaining the g-torus $\mathbb{T}_g$ with Euler characteristic $\chi(\mathbb{T}_g) = 2 - 2g$, while non-orientable surfaces are obtained by adding $h > 0$ *cross-caps* to the sphere, hence obtaining a non-orientable surface $\mathbb{P}_h$ with Euler characteristic $\chi(\mathbb{P}_h) = 2 - h$. We denote by $\overline{\Sigma}$ the surface (without boundary) obtained from Σ by gluing a disk on each of the $\beta(\Sigma)$ components of the boundary. It is then easy to show that $\chi(\overline{\Sigma}) = \beta(\Sigma) + \chi(\Sigma)$. A subset Π of a surface Σ is *surface-separating* if $\Sigma \setminus \Pi$ has at least 2 connected components. It is convenient to work with the *Euler genus* $\gamma(\Sigma)$ of a surface Σ, which is defined as $\gamma(\Sigma) = 2 - \chi(\Sigma)$.

Graphs embedded in surfaces. For a graph G we use the notation (G, τ) to denote that τ is an embedding of G in Σ, whenever the surface Σ is clear from the context. An embedding has *vertices*, *edges*, and *faces*, which are 0, 1, and 2 dimensional open sets, and are denoted $V(G)$, $E(G)$, and $F(G)$, respectively. In a *2-cell embedding*, also called *map*, each face is homeomorphic to a disk. The degree $\mathrm{d}(v)$ of a vertex v is the number of edges incident with v, counted with multiplicity (loops are counted twice). An edge of a map has two ends (also called *half-edges*), and either one or two sides, depending on the number of faces which is incident with. A map is *rooted* if an edge and one of its half-edges and sides are distinguished as the root-edge, root-end and root-side, respectively.

For a graph G, the *Euler genus* of G, denoted $\gamma(G)$, is the smallest Euler genus among all surfaces in which G can be embedded. An *O-arc* is a subset of Σ homeomorphic to $\mathbb{S}^1$. A subset of Σ meeting the drawing only at vertices of G is called *G-normal*. If an O-arc is G-normal, then we call it a *noose*. The *length* of a noose is the number of its vertices. Many results in topological graph theory rely on the concept of *representativity* [19,17], also called *face-width*, which is a parameter that quantifies local planarity and density of embeddings. The representativity $\mathbf{rep}(G, \tau)$ of a graph embedding (G, τ) is the smallest length of a non-contractible (i.e., non null-homotopic) noose in Σ. We call an embedding (G, τ) *polyhedral* [16] if G is 3-connected and $\mathbf{rep}(G, \tau) \geq 3$, or if G is a clique and $1 \leq |V(G)| \leq 3$. With abuse of notation, we also say in that case that the graph G itself is polyhedral.

For a given embedding (G, τ), we denote by (G^*, τ) its dual embedding. Thus G^* is the geometric dual of G. Each vertex v (resp. face r) in (G, τ) corresponds to some face v^* (resp. vertex r^*) in (G^*, τ). Also, given a set $S \subseteq E(G)$, we denote by S^* the set of the duals of the edges in S. Let (G, τ) be an embedding and let (G^*, τ) be its dual. We define the *radial graph embedding* (R_G, τ) of (G, τ) (also known as *vertex-face graph embedding*) as follows: R_G is an embedded bipartite graph with vertex set $V(R_G) = V(G) \cup V(G^*)$. For each pair $e = \{v, u\}$, $e^* = \{u^*, v^*\}$ of dual edges in G and G^*, R_G contains edges $\{v, v^*\}$, $\{v^*, u\}$, $\{u, u^*\}$, and $\{u^*, v\}$. The *medial graph embedding* (M_G, τ) of (G, τ) is the dual embedding of the radial embedding (R_G, τ) of (G, τ). Note that (M_G, τ) is a Σ-embedded 4-regular graph.

Tree-like decompositions of graphs. Let G be a graph on n vertices. A *branch decomposition* (T, μ) of a graph G consists of an unrooted ternary tree T (i.e., all internal vertices are of degree three) and a bijection $\mu : L \to E(G)$ from the set L of leaves of T to the edge set of G. We define for every edge e of T the *middle set* $\mathbf{mid}(e) \subseteq V(G)$ as follows: Let T_1 and T_2 be the two connected components of $T \setminus \{e\}$. Then let G_i be the graph induced by the edge set $\{\mu(f) : f \in L \cap V(T_i)\}$ for $i \in \{1, 2\}$. The *middle set* is the intersection of the vertex sets of G_1 and G_2, i.e., $\mathbf{mid}(e) := V(G_1) \cap V(G_2)$. The *width* of (T, μ) is the maximum order of the middle sets over all edges of T, i.e., $\mathbf{w}(T, \mu) := \max\{|\mathbf{mid}(e)| : e \in T\}$. An optimal branch decomposition of G is defined by a tree T and a bijection μ which give the minimum width, the *branchwidth*, denoted by $\mathbf{bw}(G)$.

Let $G = (V, E)$ be a connected graph. For $S \subseteq V$, we denote by $\delta(S)$ the set of all edges with an end in S and an end in $V \setminus S$. Let $\{V_1, V_2\}$ be a partition of V. If $G[V \setminus V_1]$ and $G[V \setminus V_2]$ are both non-null and connected, we call $\delta(V_1)$ a *bond* of G [19].

A *carving decomposition* (T, μ) is similar to a branch decomposition, only with the difference that μ is a bijection between the leaves of the tree and the vertex set of the graph G. For an edge e of T, the counterpart of the middle set, called the *cut set* $\mathbf{cut}(e)$, contains the edges of G with endvertices in the leaves of both subtrees. The counterpart of branchwidth is *carvingwidth*, and is denoted by $\mathbf{cw}(G)$. In a *bond carving decomposition*, every cut set is a bond of the graph. That is, in a bond carving decomposition, every cut set separates the graph into two connected components.

3 Polyhedral Decompositions

We introduce in this section *polyhedral decompositions* of graphs embedded in surfaces. Let G be an embedded graph, and let N be a noose in the surface. Similarly to [5], we use the notation $G \quad N$ for the graph obtained by cutting G along the noose N and gluing a disk on the obtained boundaries.

Definition 1. *Given a graph $G = (V, E)$ embedded in a surface of Euler genus γ, a* polyhedral decomposition *of G is a set of graphs $\mathcal{G} = \{H_1, \ldots, H_\ell\}$ together with a set of vertices $A \subseteq V$ such that*

- $|A| = \mathcal{O}(\gamma)$;
- H_i *is a minor of* $G[V \setminus A]$, *for* $i = 1, \ldots, \ell$;
- H_i *has a polyhedral embedding in a surface of Euler genus at most* γ, *for* $i = 1, \ldots, \ell$; *and*
- $G[V \setminus A]$ *can be constructed by joining the graphs of* $\mathcal{G}$ *applying clique sums of size* $0, 1,$ *or* 2.

Algorithm 1 provides an efficient way to construct a polyhedral decomposition, as it is stated in Proposition 1.

Algorithm 1. Construction of a polyhedral decomposition of an embedded graph

Input: A graph G embedded in a surface of Euler genus γ.
Output: A polyhedral decomposition of G.

$A = \emptyset$, $\mathcal{G} = \{G\}$ (the elements in $\mathcal{G}$, which are embedded graphs, are called *components*).
while $\mathcal{G}$ contains a non-polyhedral component H **do**
  Let N be a noose in the surface in which H is embedded,
  and let $S = V(H) \cap N$.
  if N is non-surface-separating **then**
    Add S to A, and replace in $\mathcal{G}$ component H with $H[V(H) \setminus S]$ N.
  if N is surface-separating **then**
    Let H_1, H_2 be the subgraphs of H N corresponding to the two surfaces occurring after splitting H
    if $S = \{u\} \cup \{v\}$ and $\{u, v\} \notin E(H)$ **then**
      Add the edge $\{u, v\}$ to H_i, $i = 1, 2$.
    Replace in $\mathcal{G}$ component H with the components of H N containing at least one edge of H.
return $\{\mathcal{G}, A\}$.

In the above algorithm, the addition of an edge $\{u, v\}$ represents the existence of a path in G between u and v that is not contained in the current component.

Proposition 1. *Given a graph* G *on* n *vertices embedded in a surface, Algorithm 1 constructs a polyhedral decomposition of* G *in* $\mathcal{O}(n^3)$ *steps.*

4 Surface Cut Decompositions

In this section we generalize sphere cut decompositions to graphs on surfaces; we call them *surface cut decompositions*. First we need a topological definition. A subset Π of a surface Σ is *fat-connected* if for every two points $p, q \in \Pi$, there exists a path $P \subseteq \Pi$ such that for every $x \in P$, $x \neq p, q$, there exists a subset D homeomorphic to an open disk such that $x \in D \subseteq \Pi$. We can now define the notion of surface cut decomposition.

Definition 2. *Given a graph* G *embedded in a surface* Σ, *a* surface cut decomposition *of* G *is a branch decomposition* (T, μ) *of* G *such that, for each edge* $e \in E(T)$, *there is a subset of vertices* $A_e \subseteq V(G)$ *with* $|A_e| = \mathcal{O}(\gamma(\Sigma))$ *and either*

- $|\mathbf{mid}(e) \setminus A_e| \leq 2$, *or*
- *there exists a polyhedral decomposition* $\{\mathcal{G}, A\}$ *of* G *and a graph* $H \in \mathcal{G}$ *such that*
  - $A_e \subseteq A$;
  - $\mathbf{mid}(e) \setminus A_e \subseteq V(H)$;
  - *the vertices in* $\mathbf{mid}(e) \setminus A_e$ *are contained in a set* $\mathcal{N}$ *of* $\mathcal{O}(\gamma(\Sigma))$ *nooses, such that the total number of occurrences in* $\mathcal{N}$ *of the vertices in* $\mathbf{mid}(e) \setminus A_e$ *is* $|\mathbf{mid}(e) \setminus A_e| + \mathcal{O}(\gamma(\Sigma))$; *and*
  - $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ *contains exactly two connected components, which are both fat-connected.*

Note that a sphere cut decomposition is a particular case of a surface cut decomposition when $\gamma = 0$, by taking $A_e = \emptyset$, $\mathcal{G}$ containing only the graph itself, and all the vertices of each middle set contained in a single noose.

We now show in Algorithm 2 how to construct a surface cut decomposition of an embedded graph. More details can be found in [18].

Algorithm 2. Construction of a surface cut decomposition of an embedded graph

Input: An embedded graph G.
Output: A surface cut decomposition of G.

Compute a polyhedral decomposition $\{\mathcal{G}, A\}$ of G, using Algorithm 1.
for each component H of $\mathcal{G}$ **do**
1. Compute a branch decomposition (T'_H, μ'_H) of H, using [2, Theorem 3.8].
2. Transform (T'_H, μ'_H) to a carving decomposition (T^c_H, μ^c_H) of the medial graph M_H.
3. Transform (T^c_H, μ^c_H) to a *bond* carving decomposition (T^b_H, μ^b_H) of M_H, using [19].
4. Transform (T^b_H, μ^b_H) to a branch decomposition (T_H, μ_H) of H.

Construct a branch decomposition (T, μ) of G by merging the branch decompositions $\{(T_H, \mu_H) \mid H \in \mathcal{G}\}$, and by adding the set of vertices A to all the middle sets.
return (T, μ).

Theorem 1. *Given a graph* G *on* n *vertices embedded in a surface of Euler genus* γ, *with* $\mathbf{bw}(G) \leq k$, *Algorithm 2 constructs, in* $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ *steps, a surface cut decomposition* (T, μ) *of* G *of width at most* $27k + \mathcal{O}(\gamma)$.

How surface cut decompositions are used for dynamic programming We shall now discuss how surface cut decompositions guarantee good upper bounds on the size of the tables of dynamic programming algorithms for packing-encodable problems. The size of the tables depends on how many ways a partial solution can intersect a middle set during the dynamic programming algorithm. The advantage of a surface cut decomposition is that the middle sets are placed on the surface in such a way that permits to give a precise asymptotic enumeration of the size of the tables. Indeed, in a surface cut decomposition, once we remove a set of vertices whose size is linearly bounded by γ, the middle sets are

either of size at most two (in which case the size of the tables is bounded by a constant) or are situated around a set of $\mathcal{O}(\gamma)$ nooses, where vertices can be repeated at most $\mathcal{O}(\gamma)$ times. In such a setting, the number of ways that a partial solution can intersect a middle set is bounded by the number of non-crossing partitions of the boundary-vertices in a fat-connected subset of the surface (see Definition 2). By splitting the boundary-vertices that belong to more than one noose, we can assume that these nooses are mutually disjoint. That way, we reduce the problem to the enumeration of the non-crossing partitions of $\mathcal{O}(\gamma)$ disjoint nooses containing ℓ vertices, which are $2^{\mathcal{O}(\ell)} \cdot \ell^{\mathcal{O}(\gamma)} \cdot \gamma^{\mathcal{O}(\gamma)}$, as we prove in the following section (Theorem 3). Observe that the splitting operation increases the size of the middle sets by at most $\mathcal{O}(\gamma)$, therefore $\ell = k + \mathcal{O}(\gamma)$ and this yields an upper bound of $2^{\mathcal{O}(k)} \cdot k^{\mathcal{O}(\gamma)} \cdot \gamma^{\mathcal{O}(\gamma)}$ on the size of the tables of the dynamic programming. In Section 5 we use singularity analysis over expressions obtained by the symbolic method to count the number of such non-crossing partitions. Namely, in Sections 5.1 and 5.2 we give a precise estimate of the number of non-crossing partitions in surfaces with boundary. Then we incorporate two particularities of surface cut decompositions: firstly, we deal with the set A of vertices originating from the polyhedral decomposition. These vertices are not situated around the nooses that disconnect the surface into two connected components, and this is why they are treated as *apices* in the enumeration. Secondly, we take into account that, in fact, we need to count the number of non-crossing *packings* rather than the number of non-crossing partitions, as a solution may not intersect *all* the vertices of a middle set, but only a subset. The combinatorial results of Section 5 are of interest by themselves, as they are a natural extension to higher-genus surfaces of the classical non-crossing partitions in the plane, which are enumerated by the Catalan numbers (see e.g. [14]).

5 Non-crossing Partitions in Surfaces with Boundary

In this section we obtain upper bounds for non-crossing partitions in surfaces with boundary. The concept of a non-crossing partition in a general surface is not as simple as in the case of the disk, and must be defined carefully. In Section 5.1 we set up our notation. In Section 5.2 we obtain a tree-like structure that provides a way to obtain asymptotic estimates. In this part, we exploit map enumeration techniques, together with singularity analysis.

5.1 2-Zone Decompositions and Non-crossing Partitions

Let Σ be a surface with boundary. A 2-*zone decomposition of* Σ is a decomposition of Σ where all vertices lay in the boundary of Σ and there is a coloring of the faces using 2 colors (black and white) such that every vertex is incident (possibly more than once) with a unique black face. Black faces are also called *blocks*. A 2-zone decomposition is *regular* if every block is contractible. All 2-zone decompositions are *rooted*: every connected component of the boundary of Σ is edge-rooted. We denote by $\mathcal{S}_\Sigma(k), \mathcal{R}_\Sigma(k)$ the set of general and regular

2-zone decompositions of Σ with k vertices, respectively. A 2-zone decomposition s over Σ defines a non-crossing partition $\pi_\Sigma(s)$ over the set of vertices. Let $\Pi_\Sigma(k)$ be the set of non-crossing partitions of Σ with k vertices. The main objective of this section is to obtain bounds for $|\Pi_\Sigma(k)|$. The critical observation is that each non-crossing partition is defined by a 2-zone decomposition. Consequently, $|\Pi_\Sigma(k)| \leq |\mathcal{S}_\Sigma(k)|$. The strategy to enumerate this second set consists in reducing the enumeration to simpler families of 2-zone decompositions. More specifically, the following propositionshows that it is sufficient to study regular decompositions:

Proposition 2. *Let $s \in \mathcal{S}_\Sigma$ be a 2-zone decomposition of Σ and let $\pi_\Sigma(s)$ be the associated non-crossing partition. Then there exists a regular 2-zone decomposition $m \in \mathcal{R}_\Sigma$ such that $\pi_\Sigma(s) = \pi_\Sigma(m)$.*

In other words, $|\Pi_\Sigma(k)| \leq |\mathcal{S}_\Sigma(k)| \leq |\mathcal{R}_\Sigma(k)|$ for each value of k. Instead of counting $|\mathcal{R}_\Sigma(k)|$, we reduce our study to the family of regular 2-zone decompositions where each face (block or white face) is contractible. The reason is that, as we show later, this subfamily provides the greatest contribution to the asymptotic enumeration. This set is called the set of *irreducible* 2-zone decompositions of Σ, and it is denoted by $\mathcal{P}_\Sigma(k)$. Equivalently, an irreducible 2-zone decomposition cannot be realized in a proper surface contained in Σ. The details can be found in [18].

5.2 Tree-Like Structures, Enumeration, and Asymptotic Counting

In this subsection we provide estimates for the number of irreducible 2-zone decompositions, which are obtained directly for the surface Σ. The main point consists in exploiting tree-like structures of the dual graph associated to an irreducible 2-zone decomposition. For simplicity of the presentation, the construction is explained on the disk. The dual graph of a non-crossing partition on the disk is a tree whose internal vertices are bicolored (black color for blocks). We use this family of trees in order to obtain a decomposition of elements of the set $\mathcal{P}_\Sigma(k)$. (The reader which is not familiar with the symbolic method and analytic combinatorics is referred to [13].) In [18] the enumeration of this basic family is done, as well as the enumeration of the related families.

The construction for general surfaces is a generalization of the previous one. An example is shown in the leftmost picture of Fig. 1. For an element $m \in \mathcal{P}_\Sigma(k)$, denote by M the resulting map on $\overline{\Sigma}$ (recall the definition of $\overline{\Sigma}$ in Section 2). From M we reconstruct the initial 2-zone decomposition m by pasting vertices of degree 1 which are incident to the same face, and taking the dual map. From M we define a new rooted map on $\overline{\Sigma}$ as follows: we start deleting recursively vertices of degree 1 which are not roots. Then we continue dissolving vertices of degree 2. The resulting map has $\beta(\Sigma)$ faces and all vertices have degree at least 3 (apart from root vertices, which have degree 1). The resulting map is called the *scheme associated* to M; we denote it by S_M. See Fig. 1 for an example.

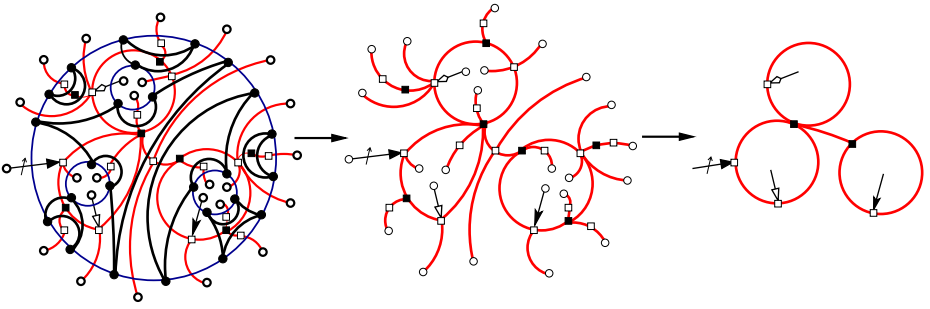


Fig. 1. Construction of the scheme of an element in $\mathcal{P}_\Sigma$. The dual of an irreducible 2-zone decomposition is shown on the left. After deleting vertices of degree 1 recursively and dissolving vertices of degree 2, we obtain the associated scheme on the right.

An inverse construction can be done using maps over $\overline{\Sigma}$ and families of plane trees. Using these basic pieces, we can reconstruct all irreducible 2-zone decompositions. Exploiting this decomposition and using singularity analysis, we get the following theorem (Γ denotes the classical Gamma function [13]):

Theorem 2. *Let Σ be a surface with boundary. Then,*

$$|\Pi_\Sigma(k)| \leq_{k\to\infty} \frac{C(\Sigma)}{\Gamma\left(3/2\gamma(\Sigma)+\beta(\Sigma)-3\right)} \cdot k^{3/2\gamma(\Sigma)+\beta(\Sigma)-4} \cdot 4^k,$$

where $C(\Sigma)$ is a function depending only on Σ that is bounded by $\gamma(\Sigma)^{\mathcal{O}(\gamma(\Sigma))}$.

Additional constructions. So far, we enumerated families of non-crossing partitions with boundary. Firstly, in surface cut decompositions we need to deal with a set of additional vertices that play the role of *apices* (cf. the last paragraph of Section 4). Secondly, we show how to extend the enumeration from non-crossing partitions to non-crossing packings. In both cases, we show that the modification over generating functions (GFs for short) does not depend on the surface Σ where non-crossing partitions are considered. The analysis consists in symbolic manipulation of GFs and application of singularity analysis over the resulting expressions. Combining the univariate asymptotic obtained in Theorem 2 with the constructions described above, we obtain the bound on the size of the tables when using surface cut decompositions:

Theorem 3. *Let $\overline{\Pi_{\Sigma,l}}(k)$ be the set of non-crossing partitions of Σ with k vertices and a set of l apices. Then the value $\sum_{i=0}^{k}\binom{k}{i}\left|\overline{\Pi_{\Sigma,l}}(k)\right|$ is upper-bounded, for large k, by*

$$\frac{C(\Sigma)}{2^{2+l}\Gamma\left(3/2\gamma(\Sigma)+\beta(\Sigma)-3\right)} \cdot k^{3/2\gamma(\Sigma)+\beta(\Sigma)-4+l} \cdot 5^{k+1},$$

where $C(\Sigma)$ is a function depending only on Σ that is bounded by $\gamma(\Sigma)^{\mathcal{O}(\gamma(\Sigma))}$.

6 Conclusions and Open Problems

Our results can be summarized as follows.

Theorem 4. *Given a packing-encodable problem P in a graph G embedded in a surface of Euler genus γ, with $\mathbf{bw}(G) \leq k$, the size of the tables of a dynamic programming algorithm to solve P on a surface cut decomposition of G is bounded above by $2^{\mathcal{O}(k)} \cdot k^{\mathcal{O}(\gamma)} \cdot \gamma^{\mathcal{O}(\gamma)}$.*

As we mentioned, the problems tackled in [9] can be encoded with pairings, and therefore they can be seen as special cases of packing-encodable problems. As a result of this, we reproduce all the results of [9]. Moreover, as our approach does not use planarization, our analysis provides algorithms where the dependence on the Euler genus γ is better than the one in [9]. In particular, the running time of the algorithms in [9] is $2^{\mathcal{O}(\gamma \cdot \mathbf{bw} + \gamma^2 \cdot \log(\mathbf{bw}))} \cdot n$, while in our case the running time is $2^{\mathcal{O}(\mathbf{bw} + \gamma \cdot \log(\mathbf{bw}) + \gamma \cdot \log \gamma)} \cdot n$.

Dynamic programming is important for the design of *subexponential* exact or parameterized algorithms. Using the fact that bounded-genus graphs have branchwidth at most $\mathcal{O}(\sqrt{\gamma \cdot n})$ [15], we derive the existence of exact algorithms in $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{\gamma n} + \gamma \cdot \log(\gamma \cdot n))})$ steps for all packing-encodable problems. Moreover, using bidimensionality theory (see [7,8]), one can derive $2^{\mathcal{O}(\gamma \cdot \sqrt{k} + \gamma \cdot \log(\gamma \cdot k))} \cdot n^{\mathcal{O}(1)}$ step parameterized algorithms for all bidimensional packing-encodable problems.

Sometimes dynamic programming demands even more complicated encodings. We believe that our results can also serve in this direction. For instance, surface cut decompositions have recently been used in [1] for minor containment problems, where tables encode partitions of packings of the middle sets.

A natural extension of our results is to consider more general classes of graphs than bounded-genus graphs. This has been done in [11] for problems where the tables of the algorithms encode pairings of the middle sets. To extend these results for packing-encodable problems (where tables encode subsets of the middle sets) using the planarization approach of [11] appears to be a quite complicated task. We believe that our surface-oriented approach could be more successful in this direction and we find it an interesting, but non-trivial task.

Acknowledgement. We would like to thank Marc Noy and Sergio Cabello for valuable ideas and for pointing us to several interesting references.

References

1. Adler, I., Dorn, F., Fomin, F.V., Sau, I., Thilikos, D.M.: Faster Parameterized Algorithms for Minor Containment. In: Proc. of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT) (to appear, 2010)
2. Amir, E.: Efficient approximation for triangulation of minimum treewidth. In: Proc. of the 17th Conf. on Uncertainty in Artificial Intelligence (UAI), pp. 7–15 (2001)
3. Arnborg, S.: Efficient algorithms for combinatorial problems on graphs with bounded decomposability – a survey. BIT 25(1), 2–23 (1985)

4. Bodlaender, H.L.: Dynamic programming on graphs with bounded treewidth. In: Lepistö, T., Salomaa, A. (eds.) ICALP 1988. LNCS, vol. 317, pp. 105–118. Springer, Heidelberg (1988)
5. Cabello, S., Mohar, B.: Finding shortest non-separating and non-contractible cycles for topologically embedded graphs. Discrete and Computational Geometry 37, 213–235 (2007)
6. Courcelle, B.: The monadic second-order logic of graphs: definable sets of finite graphs. In: van Leeuwen, J. (ed.) WG 1988. LNCS, vol. 344, pp. 30–53. Springer, Heidelberg (1989)
7. Demaine, E.D., Fomin, F.V., Hajiaghayi, M.T., Thilikos, D.M.: Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. Journal of the ACM 52(6), 866–893 (2005)
8. Demaine, E.D., Hajiaghayi, M., Thilikos, D.M.: The Bidimensional Theory of Bounded-Genus Graphs. SIAM Journal on Discrete Mathematics 20(2), 357–371 (2006)
9. Dorn, F., Fomin, F.V., Thilikos, D.M.: Fast Subexponential Algorithm for Non-local Problems on Graphs of Bounded Genus. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 172–183. Springer, Heidelberg (2006)
10. Dorn, F., Fomin, F.V., Thilikos, D.M.: Subexponential parameterized algorithms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 15–27. Springer, Heidelberg (2007)
11. Dorn, F., Fomin, F.V., Thilikos, D.M.: Catalan structures and dynamic programming in H-minor-free graphs. In: Proc. of the 19th annual ACM-SIAM Symposium on Discrete algorithms (SODA), pp. 631–640 (2008)
12. Dorn, F., Penninkx, E., Bodlaender, H.L., Fomin, F.V.: Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Branch Decompositions. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 95–106. Springer, Heidelberg (2005)
13. Flajolet, F., Sedgewick, R.: Analytic Combinatorics. Cambridge University Press, Cambridge (2008)
14. Flajolet, P., Noy, M.: Analytic combinatorics of non-crossing configurations. Discrete Mathematics 204(1), 203–229 (1999)
15. Fomin, F.V., Thilikos, D.M.: Fast Parameterized Algorithms for Graphs on Surfaces: Linear Kernel and Exponential Speed-Up. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 581–592. Springer, Heidelberg (2004)
16. Mohar, B., Thomassen, C.: Graphs on surfaces. John Hopk. Univ. Press, Baltimore (2001)
17. Robertson, N., Seymour, P.: Graph minors. XII. Distance on a surface. J. Combin. Theory Series B 64, 240–272 (1995)
18. Rué, J., Sau, I., Thilikos, D.M.: Dynamic Programming for Graphs on Surfaces. Research Report RR-7166, INRIA (2009), hal.archives-ouvertes.fr/inria-00443582
19. Seymour, P., Thomas, R.: Call routing and the ratcatcher. Combinatorica 14(2), 217–241 (1994)
20. Telle, J.A., Proskurowski, A.: Algorithms for vertex partitioning problems on partial k-trees. SIAM Journal on Discrete Mathematics 10(4), 529–550 (1997)

Interval Graphs: Canonical Representation in Logspace

Johannes Köbler[1], Sebastian Kuhnert[1], Bastian Laubner[1], and Oleg Verbitsky[2,⋆]

[1] Humboldt-Universität zu Berlin
[2] Institute for Applied Problems of Mechanics and Mathematics, Ukrainian Academy of Sciences, Lviv
{koebler,kuhnert,laubner,verbitsk}@informatik.hu-berlin.de

Abstract. We present a logspace algorithm for computing a canonical labeling, in fact a canonical interval representation, for interval graphs. As a consequence, the isomorphism and automorphism problems for interval graphs are solvable in logspace.

1 Introduction

There has been persistent interest in the algorithmic aspects of interval graphs in the past decades, also spurred by their applicability to DNA sequencing (cf. [23]) and scheduling problems (cf. [17]). In 1976, Booth and Lueker presented the first recognition algorithm for interval graphs [2] running in time linear in the number of vertices and edges, which they followed up by a linear-time interval graph isomorphism algorithm [16]. These algorithms are based on a special data structure called *PQ-trees*. By pre-processing the graph's modular decomposition tree, Hsu and Ma [8] later presented a simpler linear-time recognition algorithm that avoids the use of PQ-trees. Habib et al. [6] achieve the same time bound employing the lexicographic breadth-first search of Rose, Tarjan, and Lueker [20] and in combination with smart pivoting. A parallel NC^2 algorithm was given by Klein in [10].

All of the above algorithms have in common that they compute a *perfect elimination ordering* (peo) of the graph's vertices. This ordering has the property that for every vertex, its neighborhood among its successors in the ordering forms a clique. Fulkerson and Gross [5] show that a graph has a peo if and only if it is chordal, and the above methods determine whether a graph is an interval graph once its peo has been determined in linear time.

Our methods are optimized for space complexity. As such, our exposition neither relies on computing the graph's peo, nor do we use transitive orientation algorithms for comparability graphs as in [13]. Instead, the basis of our work is the observation of Laubner [14] that in an interval graph, the graph's maximal cliques and a modular decomposition tree are definable by means of first-order logic. This makes these objects tractable in logarithmic space and leads us to

⋆ Supported in part by the Alexander von Humboldt foundation.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 384–395, 2010.

the first logspace algorithm for computing a canonical interval representation for any interval graph (note that recognition of interval graphs in L follows from the results of Reif [18]). We identify so-called *overlap components* of interval graphs whose interval representations are essentially unique and show how to compute their interval representations canonically. We color these components with their canonical interval representations and place them in a tree that allows us to combine the canonical interval representations of the components to one for the whole graph. To achieve this, we apply Lindell's algorithm [15] to the colored decomposition tree.

Finding logspace algorithms for the graph isomorphism problem of restricted graph classes is an active research area. It was started by Lindell with his canonization algorithm for trees [15]. In a series of results, Datta, Limaye, Nimbhorkar, Thierauf and Wagner generalize this to planar graphs [3], whereas Köbler and Kuhnert show the generalization to k-trees [11]. In each of these cases the isomorphism problem has a matching lower bound, i. e. it turns out to be L-complete. The graph classes considered in these results have in common that their clique size is bounded by a constant. To the best of our knowledge, our L-completeness result for interval graph isomorphism is the first for a natural class of graphs containing cliques of arbitrary size.

Organisation of the paper. Section 2 introduces some preliminaries, notably the decomposition of interval graphs into overlap components. In Section 3 we show how to compute a canonical interval representation for a single overlap component in logspace. Section 4 contains our main result: We give a logspace algorithm to obtain a canonical interval representation of arbitrary interval graphs. In Section 5 we show that recognition and isomorphism testing of interval graphs is hard for logspace, thereby proving L-completeness for both problems.

2 Preliminaries

As usual, L is the class of all languages decidable by Turing machines with a read-only input tape using only $\mathcal{O}(\log n)$ bounded space on the working tapes. FL is the class of all functions computable by such machines that additionally have a write-only output tape. For a set S, we denote its cardinality by $\|S\|$.

2.1 Graphs and Set Systems

We write $G \cong H$ to say that G and H are isomorphic graphs. The vertex set of a graph G is denoted by $V(G)$. The set of all vertices having distance at most 1 from a vertex $v \in V(G)$ is called its *neighborhood* and denoted by $N(v)$. Note that $v \in N(v)$. We also use $N(u, v) = N(u) \cap N(v)$ for the common neighborhood of two vertices u and v. If $N(u) = N(v)$, we call these vertices *twins* (note that only adjacent vertices can be twins).

Let $\mathcal{F}$ be a family of sets, which will also be called a *set system*. We allow $A = B$ for some $A, B \in \mathcal{F}$, i. e. $\mathcal{F}$ is a multiset whose elements are sets. The *support* of $\mathcal{F}$ is defined by $\operatorname{supp}(\mathcal{F}) = \bigcup_{X \in \mathcal{F}} X$. Sometimes it will be useful

to regard $\mathcal{F}$ as a hypergraph on $\mathrm{supp}(\mathcal{F})$ (possibly with multiple hyperedges) and speak of isomorphic set systems. A *slot* is an inclusion-maximal subset S of $\mathrm{supp}(\mathcal{F})$ such that each $A \in \mathcal{F}$ contains either all of S or none of it.

The *intersection graph* of $\mathcal{F}$ is the graph $\mathbb{I}(\mathcal{F})$ with vertex set $\mathcal{F}$ where A and B are adjacent if they have a nonempty intersection. Note that, if $A = B$, these two vertices are twins in the intersection graph. We say that sets A and B *overlap* and write $A \between B$ if A and B have a nonempty intersection but neither of them includes the other. The *overlap graph* $\mathbb{O}(\mathcal{F})$ is a spanning subgraph of $\mathbb{I}(\mathcal{F})$ where the sets A and B are adjacent iff they overlap or are equal.

Of course, $\mathbb{O}(\mathcal{F})$ can be disconnected even if $\mathbb{I}(\mathcal{F})$ is connected. Subsets of $\mathcal{F}$ that span a connected component of $\mathbb{O}(\mathcal{F})$ will be referred to as *overlap components* of $\mathcal{F}$. Note that overlap components are set systems rather than subgraphs of $\mathbb{O}(\mathcal{F})$ and thus contain more information. If $\mathcal{O}$ and $\mathcal{O}'$ are different overlap components, then either every two sets $A \in \mathcal{O}$ and $A' \in \mathcal{O}'$ are disjoint or all sets of one of the two components are contained in a single slot of the other component. This containment relation determines a tree-like decomposition of $\mathcal{F}$ into its overlap components.

We denote *intervals* as $[a, b] = \{i \in \mathbb{N}_0 \mid a \leq i \leq b\}$. We say $[a_1, b_1] < [a_2, b_2]$ if $a_1 < a_2$ or if $a_1 = a_2$ and $b_1 < b_2$. For interval systems $\mathcal{I}$ and $\mathcal{J}$, we write $\mathcal{I} < \mathcal{J}$ if the smallest uncommon interval (with due regard to the multiplicities) belongs to $\mathcal{I}$. A graph G is an *interval graph* if it is isomorphic to the intersection graph of a family of intervals $\mathcal{I}$. Such an isomorphism $\ell : V(G) \to \mathcal{I}$ is called an *interval labeling* of G. The interval system $\mathcal{I}$ is called *interval representation* of G and will also be denoted by G^ℓ. We call G^ℓ a *minimal* interval representation of G, if the size of $\mathrm{supp}(G^\ell)$ is as small as possible. A *canonical interval labeling* is a function that for any interval graph G produces an interval labeling ℓ_G so that $G^{\ell_G} = H^{\ell_H}$ whenever $G \cong H$.

Similarly, we call an interval system $\mathcal{I}$ an interval representation of a set system $\mathcal{F}$ if $\mathcal{I} \cong \mathcal{F}$ as hypergraphs, where we assume $\mathrm{supp}(\mathcal{I}) = [0, \|\mathrm{supp}(\mathcal{F})\| - 1]$; an interval labeling of $\mathcal{F}$ is a function $\ell\colon \mathcal{F} \to \mathcal{I}$.

2.2 Bundles of maxcliques

An inclusion-maximal clique in a graph G will be called a *maxclique*. The *(maxclique) bundle* B_v at a vertex v consists of all maxcliques containing v. Let $\mathcal{B}_G = \{B_v\}_{v \in V(G)}$. The *bundle hypergraph* of G has the maxcliques of G as vertices and the bundles in $\mathcal{B}_G$ as hyperedges. By the overlap components of G we mean the overlap components of $\mathcal{B}_G$.

Lemma 1. *Every maxclique C of an interval graph G contains vertices u and v such that $C = N(u, v)$.*

Proof. Given an interval representation of G, we will use notation I_v for the interval corresponding to a vertex $v \in V(G)$. We have $C \subseteq N(u, v)$ for any $u, v \in C$ and, therefore, we only need to find u, v such that $N(u, v) \subseteq C$. For this purpose, consider an interval representation of G and choose $u, v \in C$ so

that $I_u \cap I_v$ is inclusion-minimal. For any $w \in C$, we have $I_w \supseteq I_u \cap I_v$ for else $I_u \cap I_w$ or $I_w \cap I_v$ would be strictly included in $I_u \cap I_v$. Suppose now that $z \in N(u, v)$. Since I_z intersects $I_u \cap I_v$, it has nonempty intersection with I_w for each $w \in C$. By maximality, $z \in C$. □

This shows that maxcliques can be represented by a pair of vertices u and v such that $N(u, v)$ is a clique. A bundle B_v can be represented by the corresponding vertex v. The binary relations $B_u \subseteq B_v$ and $B_u \between B_v$ between bundles become first-order definable (in terms of the adjacency and equality relations on $V(G)$) and, therefore, decidable in logspace. Moreover, the overlap components of G can be computed in logspace using undirected reachability in $\mathbb{O}(\mathcal{B}_G)$ [19].

The following lemma implies that the bundle hypergraph contains all information on the isomorphism type of G (in fact, $G \cong \mathbb{I}(\mathcal{B}_G)$ holds for any graph). We omit the proof because of space constraints.

Lemma 2. *For every minimal interval representation $\mathcal{I}$ of an interval graph G, $\mathcal{I}$ viewed as a hypergraph is isomorphic to the bundle hypergraph of G.* □

This shows that the minimal interval representation of G is unique up to hypergraph isomorphism. We will use this fact to construct a minimal interval representation.

3 Canonizing Overlap Components

In this section we show how to compute a canonical interval labeling of overlap components of interval graphs. It is canonical in the sense that overlap components which are isomorphic as set systems are mapped to the same interval representation. Let $\mathcal{O}$ be an overlap component of an interval graph G. We call two maxcliques $M, M' \in \mathrm{supp}(\mathcal{O})$ *indistinguishable in* $\mathcal{O}$ and write $M \sim_{\mathcal{O}} M'$, if there is no bundle $B_v \in \mathcal{O}$ that contains exactly one of them. Clearly, $\sim_{\mathcal{O}}$ is an equivalence relation on $\mathrm{supp}(\mathcal{O})$. The equivalence classes of $\sim_{\mathcal{O}}$ are the slots of $\mathcal{O}$. We can assume that there are no twins in $\mathcal{O}$, as our algorithm also handles colored graphs and twins can be replaced by a single node colored with their multiplicity. If $\mathcal{O}$ consists of a single bundle B_v, we use the interval labeling $\ell_{\mathcal{O}}$ that maps B_v to $[0, \|B_v\| - 1]$. So from now on we will assume that $\mathcal{O}$ consists of at least two bundles.

The key observation for obtaining the canonical interval labeling is that the interval representation of an overlap component $\mathcal{O}$ (without twins) is unique up to reversing. We call a slot S of an overlap component $\mathcal{O}$ a *side-slot*, if each interval labeling of $\mathcal{O}$ places it at the left or right end. Using only the structure of $\mathcal{O}$, we first show that the two side-slots of $\mathcal{O}$ can be identified; then we show how to compute the order on the other slots once the left end is fixed.

Lemma 3. *If an overlap component $\mathcal{O}$ of an interval graph consists of at least two bundles that are no twins, then it has exactly two side-slots. These slots can be found in* FL.

Notice that a slot $S = [M]_{\sim_{\mathcal{O}}}$ can be represented by a triple (u, v, w), where $M = N(u, v)$ is any maxclique contained in S and B_w is any bundle in $\mathcal{O}$.

Proof sketch. Let ℓ be any interval labeling of $\mathcal{O}$ (it exists by Lemma 2). We call a bundle B *marginal* if its intersections with the overlapping bundles $B' \between B$ form a single inclusion chain. We can identify the two bundles $B_1, B_2 \in \mathcal{O}$ that are mapped to the longest interval starting leftmost and the longest interval ending rightmost: (1) They are marginal, and (2) they are no subset of any other bundle of $\mathcal{O}$. (Other bundles are overlapped from both sides, yielding two inclusion chains, or spanned by a larger bundle; otherwise $\mathcal{O}$ would not be overlap-connected.)

We now choose two maxcliques M_1 and M_2 representing the two side-slots of $\mathcal{O}$: Let $\mathcal{B}_i$ be the set of marginal bundles that are contained in B_i (including B_i itself; this excludes marginal bundles at the other side). Choose $M_i \in B_i$ such that it is not contained in any bundle $B \in \mathcal{O} \setminus \mathcal{B}_i$ and in as few $B \in \mathcal{B}_i$ as possible. This construction uses only information available in $\mathcal{O}$ and is possible in logspace. Its correctness is readily verified using the isomorphism to $\mathcal{O}^\ell$. □

Following [14], we can now use a side-slot S to define a partial order $\prec_S$ on $\operatorname{supp}(\mathcal{O})$ as the smallest relation that satisfies the following properties:

1. $C \prec_S D$ for each $C \in S$ and $D \notin S$.
2. For each bundle B_v in $\mathcal{O}$ and maxcliques $C_1, C_2 \in B_v$, $D \notin B_v$:

$$C_1 \prec_S D \Leftrightarrow C_2 \prec_S D \qquad \text{and} \qquad D \prec_S C_1 \Leftrightarrow D \prec_S C_2 \tag{1}$$

$C \prec_S D$ can be read as "if slot S is leftmost, then maxclique C is left of D".

Lemma 4. *If S is a side-slot of $\mathcal{O}$, then $\prec_S$ induces a strict linear order on the slots of $\mathcal{O}$.*

Proof. Consider any interval labeling of $\mathcal{O}$. By Lemma 3 we can assume that S is placed leftmost (reverse the interval representation if necessary). Since the equivalences in (1) are true for the ordering on the maxcliques induced by any interval representation, $\prec_S$ is a subrelation of a strict linear order and is therefore an asymmetric relation on the maxcliques in $\operatorname{supp}(\mathcal{O})$. Moreover, from the definition of $\prec_S$ it is clear that any two indistinguishable maxcliques $C, D \in \operatorname{supp}(\mathcal{O})$ are incomparable w. r. t. $\prec_S$.

Now let $C, D \in \operatorname{supp}(\mathcal{O})$ with $C \not\sim_{\mathcal{O}} D$. We claim that either $C \prec_S D$ or $D \prec_S C$. Let $B_0 \between \cdots \between B_k$ be a shortest overlap-path in $\mathcal{O}$ such that $S \subseteq B_0$ and B_k contains precisely one of C and D. The claim is proved by induction on the length k of the path. If $k = 0$, the claim clearly holds. If $k \geq 1$, suppose w. l. o. g. that $C \in B_k$. If $D \notin B_{k-1}$ then for any $E \in B_{k-1}$ we have either $D \prec_S E$ or $E \prec_S D$ by induction, and since $B_{k-1} \cap B_k \neq \emptyset$ the claim follows. If $D \in B_{k-1}$, then also $C \in B_{k-1}$ since we assumed the path to be shortest. Then for any $F \in B_k \setminus B_{k-1}$, we have $D \prec_S F$ or $F \prec_S D$ by induction, and again the claim follows by (1). □

By Lemma 4 we can use $\prec_S$ to define an interval labeling ℓ_S of $\mathcal{O}$:

$$\begin{aligned} \ell_S \colon \mathcal{O} &\to \left\{ [l, r] \;\middle|\; l, r \in \left[0, \|\operatorname{supp}(\mathcal{O})\| - 1\right] \right\} \\ B_v &\mapsto \left[\operatorname{pos}(B_v), \operatorname{pos}(B_v) + \|B_v\| - 1 \right], \end{aligned}$$

where a maxclique $C \in \mathrm{supp}(\mathcal{O})$ has position $\mathrm{pos}(C) = \|\{D \in \mathrm{supp}(\mathcal{O}) \mid D \prec_S C\}\|$ and a bundle $B_v \in \mathcal{O}$ has position $\mathrm{pos}(B_v) = \min\{\mathrm{pos}(C) \mid C \in B_v\}$.

Let ℓ_{S_1} and ℓ_{S_2} be the interval labelings for $\mathcal{O}$ corresponding to the two side-slots S_1 and S_2 of $\mathcal{O}$. As the pair $\{\mathcal{O}^{\ell_{S_1}}, \mathcal{O}^{\ell_{S_2}}\}$ of interval representations only depends on the structure of G, it is invariant under isomorphism. Hence, we can choose a canonical interval labeling $\ell_{\mathcal{O}}$ of $\mathcal{O}$ among ℓ_{S_1} and ℓ_{S_2} by requiring that the set $\mathcal{O}^{\ell_{\mathcal{O}}} = \{\ell_{\mathcal{O}}(B_v) \mid B_v \in \mathcal{O}\}$ of intervals becomes minimal. We denote the corresponding order on the maxcliques of $\mathcal{O}$ by $\prec_{\mathcal{O}}$. By $\vec{\mathcal{O}}$ we denote the list of vertices v whose bundles B_v are in $\mathcal{O}$, ordered by their intervals $\ell_{\mathcal{O}}(B_v)$.

Lemma 5. *Given an interval graph G and an overlap component $\mathcal{O}$ of G, the following can be done in logspace:*

1. *Computing the partial order $\prec_{\mathcal{O}}$ on $\mathrm{supp}(\mathcal{O})$,*
2. *computing a canonical interval labeling $\ell_{\mathcal{O}}$ of $\mathcal{O}$,*
3. *computing the corresponding ordered list of vertices $\vec{\mathcal{O}}$, and*
4. *deciding if $\mathcal{O}^{\ell_{\mathcal{O}}}$ is mirror-symmetric or not.*

Proof. To prove that $C \prec_S D$ can be decided in logspace, we construct an undirected graph with nodes $\{(C_1, C_2) \mid C_1 \neq C_2 \text{ maxcliques in } \mathcal{O}\}$ meaning "$C_1 \prec_S C_2$" and edges corresponding to the equivalences given in the definition (1) of $\prec_S$. We also add a start vertex s and connect it to all nodes (C, D) with $C \in S$, $D \notin S$. Now we have $C \prec_S D$ iff (C, D) is reachable from s. Reachability in undirected graphs is decidable in L using Reingold's algorithm [19].

Once $\prec_S$ can be decided in logspace, it is easy to compute ℓ_S and to choose the left side-slot $S \in \{S_1, S_2\}$ such that $\mathcal{O}^{\ell_S} = \min\{\mathcal{O}^{\ell_{S_1}}, \mathcal{O}^{\ell_{S_2}}\}$. $\mathcal{O}^{\ell_{\mathcal{O}}}$ is mirror-symmetric iff both are equal.

Given $\ell_{\mathcal{O}}$, it is easy to compare the bundles in $\mathcal{O}$, and thereby compute $\vec{\mathcal{O}}$. □

4 Canonizing Interval Graphs

Let G be an interval graph. Again we assume that G has no twins, but allow a coloring of G instead. We also assume that G is connected – if not, we add a new vertex and connect it to all others. We define the position of a slot S in $\mathcal{O}$ analogously to that of a bundle, namely $\mathrm{pos}(S) = \min\{\mathrm{pos}(C) \mid C \in S\}$ (in fact, these positions are all equal). Additionally we define the position of S from the right: $\mathrm{rpos}(S) = \min_{C \in S} \|\{D \in \mathrm{supp}(\mathcal{O}) \mid C \prec_{\mathcal{O}} D\}\|$. If an overlap component $\mathcal{O}$ has a mirror-symmetric canon $\mathcal{O}^{\ell_{\mathcal{O}}}$, we call a slot S of $\mathcal{O}$ *low* if $\mathrm{pos}(S) < \mathrm{rpos}(S)$, *middle* if $\mathrm{pos}(S) = \mathrm{rpos}(S)$, and *high* if $\mathrm{pos}(S) > \mathrm{rpos}(S)$. If $\mathcal{O}^{\ell_{\mathcal{O}}}$ is not mirror-symmetric, we call all its slots *low*. We say an overlap component $\mathcal{O}'$ is *located at* a slot S (of $\mathcal{O}$), if $\mathrm{supp}(\mathcal{O}') \subseteq S$ and there is no overlap component $\mathcal{O}''$ such that $\mathrm{supp}(\mathcal{O}') \subset \mathrm{supp}(\mathcal{O}'') \subset \mathrm{supp}(\mathcal{O})$ (note that different overlap components have different supports).

4.1 Tree Representation

Using the above notions, we now define a tree representation for interval graphs.

Definition 1. *For a connected interval graph G without twins, its* tree representation $\mathbb{T}(G)$ *is defined by*

$$\begin{aligned} V(\mathbb{T}(G)) =& \{\vec{\mathcal{O}}, lo_{\mathcal{O}}, mi_{\mathcal{O}}, hi_{\mathcal{O}} \mid \mathcal{O} \text{ is an overlap component of } G\} \\ & \cup \{S \mid S \text{ is a slot of some overlap component } \mathcal{O} \text{ of } G\} \\ E(\mathbb{T}(G)) =& \{(\vec{\mathcal{O}}, lo_{\mathcal{O}}), (\vec{\mathcal{O}}, mi_{\mathcal{O}}), (\vec{\mathcal{O}}, hi_{\mathcal{O}}) \mid \mathcal{O} \text{ is an overlap component of } G\} \\ & \cup \{(lo_{\mathcal{O}}, S), (mi_{\mathcal{O}}, S), (hi_{\mathcal{O}}, S) \mid S \text{ is a low/middle/high slot in } \mathcal{O}\} \\ & \cup \{(S, \vec{\mathcal{O}}) \mid \text{the overlap component } \mathcal{O} \text{ is located at slot } S\} \end{aligned}$$

Further we define a coloring c of the component-nodes $\vec{\mathcal{O}}$ and slot-nodes S by

$$c(\vec{\mathcal{O}}) = \mathcal{O}^{\ell o}$$

$$c(S) = \begin{cases} \mathrm{pos}(S) & \text{if } S \text{ is low or middle} \\ \mathrm{rpos}(S) & \text{if } S \text{ is high} \end{cases}$$

If G is colored, the intervals in $c(\vec{\mathcal{O}}) = \mathcal{O}^{\ell o}$ inherit the colors of the corresponding vertices in $V(G)$.

As G is connected, there is an overlap component $\mathcal{O}_0$ such that all maxcliques of G belong to $\mathrm{supp}(\mathcal{O}_0)$. $\vec{\mathcal{O}}_0$ is the root of the directed tree $\mathbb{T}(G)$.

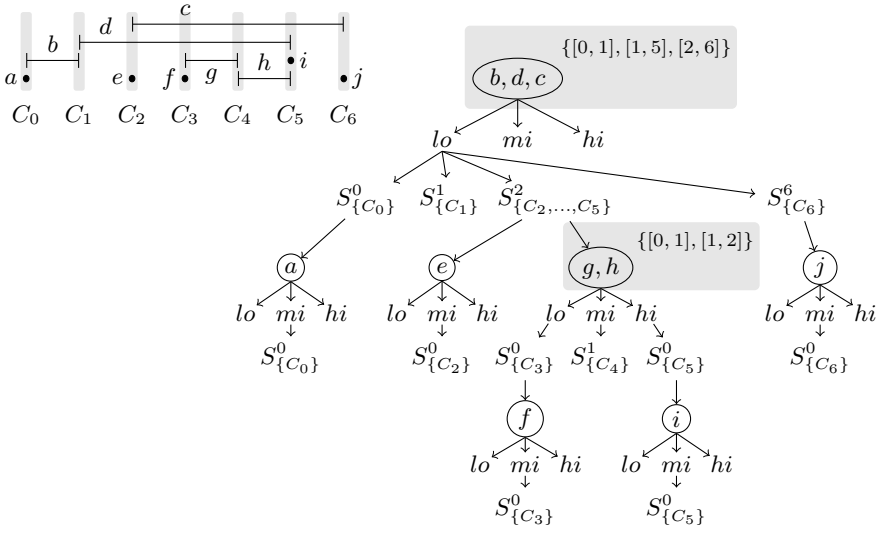


Fig. 1. An interval graph representation of a graph G and the corresponding tree representation $\mathbb{T}(G)$. Gray rectangles in $\mathbb{T}(G)$ indicate the color of overlap components. Overlap components have color $\{[0, 0]\}$ where not indicated. Slot name $S^k_{\{C_i,\ldots,C_j\}}$ denotes slot $\{C_i, \ldots, C_j\}$ and indicates that its color is k. We omit the indices of the *lo*, *mi* and *hi* nodes as they are clear from the structure of the tree.

Our goal is to compute a canonical interval labeling of G using a modified version of Lindell's canonization algorithm for trees [15] on $\mathbb{T}(G)$.

It can easily be verified that $\mathbb{T}(G)$ can be computed in logspace. We proceed to show a basic structural property of $\mathbb{T}(G)$.

Lemma 6. *There is a one-to-one correspondence between maxcliques of G and the leaf-nodes of $\mathbb{T}(G)$ that are slots.*

Proof. Let S be a slot of some overlap component $\mathcal{O}$ such that S is a leaf of $\mathbb{T}(G)$. By definition of slots, the maxcliques in S are not distinguished by $\mathcal{O}$. The bundles in the other overlap components cannot distinguish them either, as this would imply an overlap path to the bundles of $\mathcal{O}$ spanning S. So there is no vertex $v \in V(G)$ whose bundle B_v contains only a part of S, and S must be a single maxclique of G.

For the other direction, let M be any maxclique of G. Then $M \in \mathrm{supp}(\mathcal{O}_0)$, as the root overlap component $\mathcal{O}_0$ contains all maxcliques. Now if $M \in \mathrm{supp}(\mathcal{O})$ for some overlap component $\mathcal{O}$, then M is in exactly one of the slots of $\mathcal{O}$, as slots are equivalence classes and thus partition $\mathrm{supp}(\mathcal{O})$. And if M is contained in some slot S that is not a leaf in $\mathbb{T}(G)$, M must be contained in exactly one overlap component located at S (if none, M would not be maximal, if more, these overlap components would overlap). Using these observations, we can trace M to a single slot S that is a leaf of $\mathbb{T}(G)$. □

We will compute a canonical labeling of $\mathbb{T}(G)$ and call it $\ell_{\mathbb{T}(G)}$. For this, we observe a generalization of Lindell's tree canonization algorithm [15].

Lemma 7. *Lindell's algorithm [15] can be extended to colored trees and to output not only a canonical form, but also a canonical labeling. This modification preserves the logarithmic space bound.*

Proof sketch. Colors can be handled by extending the *tree isomorphism order* defined in [15] by using $color(s) < color(t)$ as additional condition (where s and t are the roots of the trees to compare). The canonical labeling can be computed by using a counter i initialized to 0: Instead of printing (the opening parenthesis of) the canon of a node v, increment i and print "$v \mapsto i$". □

4.2 Computing a Canonical Interval Labeling

Our aim is a traversal of $\mathbb{T}(G)$ that is left-to-right in the resulting canon. That is, we visit the leaf slots in ascending order of the positions of their corresponding maxcliques in the computed canonical interval representation. To achieve this, we use the canonical labeling $\ell_{\mathbb{T}(G)}$.

We first recall the *logspace tree traversal* that is used in Lindell's canonization algorithm. Only the current node must be remembered, because when given a node, it is possible in logspace to (1) go to its first child, (2) go to its next sibling, and (3) go to its parent. "First" and "next" can be respective to any order on the children of a node that can be evaluated in logspace. In our left-to-right traversal we use the following order:

- The children of an overlap component node $\vec{\mathcal{O}}$ are either ordered $lo_{\mathcal{O}} < mi_{\mathcal{O}} < hi_{\mathcal{O}}$ (if $\mathcal{O}^{\ell_{\mathcal{O}}}$ is not mirror-symmetric or if $\ell_{\mathbb{T}(G)}(lo_{\mathcal{O}}) < \ell_{\mathbb{T}(G)}(hi_{\mathcal{O}})$) or $hi_{\mathcal{O}} < mi_{\mathcal{O}} < lo_{\mathcal{O}}$ (otherwise).
- The children of the first child of an overlap component node $\vec{\mathcal{O}}$ (this can be either $lo_{\mathcal{O}}$ or $hi_{\mathcal{O}}$) are visited in ascending order of their colors.
- The children of the last child of an overlap component node $\vec{\mathcal{O}}$ (this can be either $hi_{\mathcal{O}}$ or $lo_{\mathcal{O}}$) are visited in descending order of their colors.
- The children of a slot node are ordered by their label assigned by $\ell_{\mathbb{T}(G)}$.

Note that the children of $lo_{\mathcal{O}}$ and $hi_{\mathcal{O}}$ all have different colors. Also, $mi_{\mathcal{O}}$ can have at most one child. All these conditions can be evaluated in logspace without using non-local information. Traversing $\mathbb{T}(G)$ in this order makes sure that the slots of an overlap component $\mathcal{O}$ are visited either in ascending or descending order of their positions. The latter case can only occur if $\mathcal{O}^{\ell_{\mathcal{O}}}$ is mirror-symmetric.

We complete the description of our algorithm by showing how, while processing $\mathbb{T}(G)$, a canonical interval labeling can be computed in logspace. Additionally to the current node we store a *current offset* o that equals to the number of maxcliques we have passed already. We initialize $o = 0$ and increment it by 1 whenever the logspace tree traversal passes a slot node that is a leaf. Whenever we enter an overlap component node $\vec{\mathcal{O}} = (v_1, \ldots, v_k)$ for the first time, we output the mappings $v_i \mapsto [l_i + o, r_i + o]$ where $[l_i, r_i]$ is the ith-smallest interval in $c(\vec{\mathcal{O}}) = \mathcal{O}^{\ell_{\mathcal{O}}}$ if $lo_{\mathcal{O}} < mi_{\mathcal{O}} < hi_{\mathcal{O}}$, and the ith-largest interval otherwise. In the first case this results in $\ell_G(v) = \ell_{\mathcal{O}}(B_v) + o$. In the second case this association is mirrored: If $\alpha\colon \mathcal{O} \to \mathcal{O}$ is the hypergraph isomorphism that reverses $\mathcal{O}$, then $\ell_G(v) = \ell_{\mathcal{O}}(\alpha(B_v)) + o$. After traversing all of $\mathbb{T}(G)$, we have output a mapping for each $v \in V(G)$. We call this mapping ℓ_G.

Lemma 8. *ℓ_G is an interval labeling of G.*

Proof. Take any $u, v \in V(G)$. Let $\mathcal{O}$ and $\mathcal{O}'$ be the overlap components containing B_u and B_v, and let o and o' be the current offsets when $\mathcal{O}$ and $\mathcal{O}'$ are first entered, respectively. If $\mathcal{O} = \mathcal{O}'$, we are done because $\ell_{\mathcal{O}}$ is an interval labeling and the offset o preserves intersection. If $\operatorname{supp}(\mathcal{O})$ and $\operatorname{supp}(\mathcal{O}')$ do not intersect, then u and v are not adjacent. W. l. o. g. assume $o < o'$. Indeed we have $o + \|\operatorname{supp}(\mathcal{O})\| \leq o'$, as the offset is advanced by one for each slot leaf in the subtree of $\vec{\mathcal{O}}$ (which correspond to the maxcliques in $\operatorname{supp}(\mathcal{O})$ by Lemma 6). As $\ell_{\mathcal{O}}(B_u) \subseteq [0, \|\operatorname{supp}(\mathcal{O})\| - 1]$ (see the definition of $\ell_{\mathcal{O}}$), $\ell(u)$ and $\ell(v)$ do not intersect. If $\operatorname{supp}(\mathcal{O})$ and $\operatorname{supp}(\mathcal{O}')$ do intersect, one must be contained in the other. W. l. o. g. assume $\operatorname{supp}(\mathcal{O}') \subset \operatorname{supp}(\mathcal{O})$ and let S be the slot of $\mathcal{O}$ in which $\operatorname{supp}(\mathcal{O}')$ is contained. If u is contained in some maxclique $M \in S$ (and thereby contained in all $M \in S$), then u and v are adjacent. By Lemma 6 and the order of our tree traversal we have $\ell_{\mathcal{O}}(B_u) + o \supseteq [o', o' + \|\operatorname{supp}(\mathcal{O}')\| - 1]$. Finally, if u is contained in no maxclique $M \in S$, then u and v are not adjacent. Also the slot leaves corresponding to the maxcliques in B_u will be processed all before or all after $\vec{\mathcal{O}}'$, so $\ell_G(v)$ and $\ell_G(u)$ do not intersect. □

In the following we prove that the interval labeling ℓ_G is canonical.

Lemma 9. *If G and H are isomorphic connected interval graphs without twins, then $\mathbb{T}(G) \cong \mathbb{T}(H)$.*

Proof. Since any isomorphism ϕ between G and H induces a unique mapping of the overlap components $\mathcal{O}$ of G to isomorphic overlap components $\mathcal{O}'$ of H, it is clear how to define an isomorphism ϕ' between $\mathbb{T}(G)$ and $\mathbb{T}(H)$ on the component-nodes $\vec{\mathcal{O}}$ of $\mathbb{T}(G)$. Further, since the canonical interval representations $\mathcal{O}^{\ell_{\mathcal{O}}}$ and $\mathcal{O}'^{\ell_{\mathcal{O}'}}$ coincide, $\vec{\mathcal{O}}$ and $\phi'(\vec{\mathcal{O}})$ indeed have the same colors.

In order to define ϕ' on the *lo*, *mi* and *hi* nodes of $\mathbb{T}(G)$, consider a component-node $\vec{\mathcal{O}} = (u_1, \ldots, u_k)$ of $\mathbb{T}(G)$. If $\mathcal{O}^{\ell_{\mathcal{O}}}$ is not mirror-symmetric, then it follows that $\vec{\mathcal{O}'} = (\phi(u_1), \ldots, \phi(u_k))$. Otherwise, it is also possible that $\vec{\mathcal{O}'} = (\phi(u_k), \ldots, \phi(u_1))$. In the first case we let $\phi'(lo_{\mathcal{O}}) = lo_{\mathcal{O}'}$, $\phi'(mi_{\mathcal{O}}) = mi_{\mathcal{O}'}$, $\phi'(hi_{\mathcal{O}}) = hi_{\mathcal{O}'}$; in the second we let $\phi'(lo_{\mathcal{O}}) = hi_{\mathcal{O}'}$, $\phi'(mi_{\mathcal{O}}) = mi_{\mathcal{O}'}$ and $\phi'(hi_{\mathcal{O}}) = lo_{\mathcal{O}'}$. Finally, since all children of a *lo*, *mi* or *hi* node have different colors there is a unique way to define ϕ' on the slot-nodes of $\mathbb{T}(G)$. Now it can be easily checked that ϕ' indeed is an isomorphism between $\mathbb{T}(G)$ and $\mathbb{T}(H)$. □

Theorem 1. *Given an interval graph G, a canonical interval labeling ℓ_G for G can be computed in* FL.

Proof. We assume that G has no twins; otherwise their multiplicity can be encoded as color and the resulting interval labeling can afterwards be extended to the original graph. For this case we have already shown that the labeling ℓ_G is computable in FL and that G^{ℓ_G} is isomorphic to G. It remains to show that the labelings ℓ_G and ℓ_H of any two isomorphic interval graphs G and H map these graphs to the same interval representation $G^{\ell_G} = H^{\ell_H}$: By Lemma 9, the colored trees $\mathbb{T}(G)$ and $\mathbb{T}(H)$ are isomorphic. Hence it follows that the canonical labelings $\ell_{\mathbb{T}(G)}$ and $\ell_{\mathbb{T}(H)}$ map these trees to the same colored tree $\mathbb{T}(G)^{\ell_{\mathbb{T}(G)}} = \mathbb{T}(H)^{\ell_{\mathbb{T}(H)}}$. Further, it is easy to see that the interval representation G^{ℓ_G} only depends on the tree $\mathbb{T}(G)^{\ell_{\mathbb{T}(G)}}$, implying that $G^{\ell_G} = H^{\ell_H}$. □

There is a standard Turing reduction of the automorphism group problem (i. e. computing a generating set of the automorphism group of a given graph) to the search version of graph isomorphism for colored graphs (cf. [7,12]). It is not hard to see that this reduction can be performed in logspace.

Corollary 1. *Computing a generating set of the automorphism group of a given interval graph, and hence computing a canonical labeling coset for a given interval graph is in* FL. *Further, the automorphism problem (i. e., deciding if a given graph has a non-trivial automorphism) for interval graphs is* L*-complete.*

5 Hardness of Interval Graph Problems

All hardness results in this section are under DLOGTIME-uniform AC^0 reductions. The proofs of both lemmas are easy reductions from the L-complete problem ORD (cf. [4]), and will be contained in the journal version of this paper.

Lemma 10. *Deciding whether a given graph is an interval graph is* L*-hard.* □

By results of Reif [18] and Reingold [19], this problem is also contained in L. Thus, interval graph recognition is L-complete.

Remark 1. The reduction used in the proof of Lemma 10 also proves that it is both L-hard to decide whether a given graph is *chordal*, and whether a graph is a *unit interval graph*. Again, by results of Reif [18] and Reingold [19], recognition of chordal graphs and of unit interval graphs is therefore L-complete.

Lemma 11. *Given an interval graph G, the problem of deciding if it has a non-trivial automorphism is* L*-hard. The same holds for the problem of deciding if two interval graphs are isomorphic.* □

6 Conclusion

Going beyond interval graphs, there are several natural graph classes that suggest an investigation whether they can similarly be handled in L. For example, circular-arc graphs generalize interval graphs as intersection graphs of arcs on a circle. Just like interval graphs, circular-arc graphs can be recognized efficiently in linear time (cf. [9]). However, while intuition suggests a reduction of circular-arc graphs to interval graphs by "cutting open" the circle that carries the graph's circular-arc representation, all known algorithms require additional techniques that are fairly specific to circular-arc graphs. One of the obstacles is that maxcliques cannot be handled as easily as in Lemma 1, since there are possibly exponentially many of them.

Another generalization of interval graphs is the class of rooted directed path graphs, i. e. intersection graphs of paths in a rooted and directed tree. While in this class, maxcliques can still be recognized in a similar way as in this paper, the recursive procedure for linearly ordering maxcliques as given in Section 3 cannot be employed in the presence of tree nodes of degree ≥ 3 (cf. [14]).

In the above paragraph, it is important that trees are rooted and directed accordingly, since otherwise the class becomes graph isomorphism-complete (cf. [1]). The same is true for boxicity-d graphs ($d \geq 2$), the intersection graphs of axis-parallel boxes in $\mathbb{R}^d$ (cf. [22]). Also, the two arguably most manifest extensions of interval graphs, chordal graphs and co-comparability graphs, are known to be isomorphism-complete. Finally, we would like to point to [21] for further graph classes for which recognition and isomorphism is not known to be in L.

Acknowledgement. We thank the anonymous referees for helpful comments and detailed suggestions on how to improve this paper.

References

1. Babel, L., Ponomarenko, I.N., Tinhofer, G.: The isomorphism problem for directed path graphs and for rooted directed path graphs. J. Algorithms 21(3), 542–564 (1996)

2. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. J. Comput. Syst. Sci. 13(3), 335–379 (1976)
3. Datta, S., Limaye, N., Nimbhorkar, P., Thierauf, T., Wagner, F.: Planar graph isomorphism is in log-space. In: CCC, pp. 203–214 (2009)
4. Etessami, K.: Counting quantifiers, successor relations, and logarithmic space. Journal of Computer and System Sciences 54(3), 400–411 (1997)
5. Fulkerson, D.R., Gross, O.A.: Incidence matrices and interval graphs. Pacific Journal of Mathematics 15(3), 835–855 (1965)
6. Habib, M., McConnell, R.M., Paul, C., Viennot, L.: Lex-BFS and partition refinement. Theor. Comput. Sci. 234(1-2), 59–84 (2000)
7. Hoffmann, C.M.: Group-Theoretic Algorithms and Graph Isomorphism. LNCS, vol. 136. Springer, Heidelberg (1982)
8. Hsu, W.-L., Ma, T.-H.: Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. SIAM J. Comput. 28(3), 1004–1020 (1999)
9. Kaplan, H., Nussbaum, Y.: A simpler linear-time recognition of circular-arc graphs. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 41–52. Springer, Heidelberg (2006)
10. Klein, P.N.: Efficient parallel algorithms for chordal graphs. SIAM J. Comput. 25(4), 797–827 (1996)
11. Köbler, J., Kuhnert, S.: The isomorphism problem for k-trees is complete for logspace. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 537–548. Springer, Heidelberg (2009)
12. Köbler, J., Schöning, U., Torán, J.: The Graph Isomorphism Problem: Its Structural Complexity. Progress in Theoretical Computer Science. Birkhäuser, Basel (1993)
13. Kozen, D., Vazirani, U.V., Vazirani, V.V.: NC algorithms for comparability graphs, interval gaphs, and testing for unique perfect matching. In: Maheshwari, S.N. (ed.) FSTTCS 1985. LNCS, vol. 206, pp. 496–503. Springer, Heidelberg (1985)
14. Laubner, B.: Capturing polynomial time on interval graphs. In: LICS (to appear, 2010)
15. Lindell, S.: A logspace algorithm for tree canonization. In: STOC 1992, pp. 400–404 (1992)
16. Lueker, G.S., Booth, K.S.: A linear time algorithm for deciding interval graph isomorphism. J. ACM 26(2), 183–195 (1979)
17. Möhring, R.H.: Graphs and Order. NATO ASI Series C, Mathematical and Physical Sciences, vol. 147, pp. 41–102. D. Reidel, Dordrecht (1984)
18. Reif, J.H.: Symmetric complementation. J. ACM 31(2), 401–421 (1984)
19. Reingold, O.: Undirected st-connectivity in log-space. In: STOC, pp. 376–385. ACM, New York (2005)
20. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput. 5(2), 266–283 (1976)
21. Spinrad, J.P.: Efficient graph representations. Field Institute Monographs, vol. 19 (2003)
22. Uehara, R.: Simple geometrical intersection graphs. In: Nakano, S.-i., Rahman, M. S. (eds.) WALCOM 2008. LNCS, vol. 4921, pp. 25–33. Springer, Heidelberg (2008)
23. Zhang, P., Schon, E.A., Fischer, S.G., Cayanis, E., Weiss, J., Kistler, S., Bourne, P.E.: An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA. Bioinformatics 10(3), 309–317 (1994)

Approximating the Partition Function of the Ferromagnetic Potts Model*

Leslie Ann Goldberg[1] and Mark Jerrum[2]

[1] Department of Computer Science, University of Liverpool, Ashton Building, Liverpool L69 3BX, United Kingdom
[2] School of Mathematical Sciences Queen Mary, University of London, Mile End Road, London E1 4NS, United Kingdom

Abstract. We provide evidence that it is computationally difficult to approximate the partition function of the ferromagnetic q-state Potts model when $q > 2$. Specifically we show that the partition function is hard for the complexity class $\#\mathrm{RH}\Pi_1$ under approximation-preserving reducibility. Thus, it is as hard to approximate the partition function as it is to find approximate solutions to a wide range of counting problems, including that of determining the number of independent sets in a bipartite graph. Our proof exploits the first order phase transition of the "random cluster" model, which is a probability distribution on graphs that is closely related to the q-state Potts model. A full version of this paper, with proofs included, is available at http://arxiv.org/abs/1002.0986.

1 Introduction

Let q be a positive integer. The q-state Potts partition function of a graph $G = (V, E)$, with uniform interactions of strength $\gamma \geq -1$ along the edges, is defined as

$$Z_{\mathrm{Potts}}(G; q, \gamma) = \sum_{\sigma: V \to [q]} \prod_{e=\{u,v\} \in E} \big(1 + \gamma\, \delta(\sigma(u), \sigma(v))\big), \tag{1}$$

where $[q] = \{1, \ldots, q\}$ is a set of q spins or colours, and $\delta(s, s')$ is 1 if $s = s'$, and 0 otherwise. The partition function is a sum over "configurations" σ which assign spins to vertices in all possible ways. Mostly we shall concentrate in this paper on the ferromagnetic situation, characterised by $\gamma > 0$. In the ferromagnetic Potts model, configurations σ with many adjacent like spins make a greater contribution to the partition function $Z_{\mathrm{Potts}}(G; q, \gamma)$ than those with few. The statistical mechanical model just described was introduced by Potts [19] and generalises the classical Ising model from two to q spins.

Definition (1) applies only when q is a positive integer. However, it transpires that, regarding q as an indeterminate, (1) defines a polynomial in q, and in this way we can make sense of the Potts partition function for non-integer q. An equivalent, but more concrete way of approaching the partition function

* This work was partially supported by the EPSRC grant *The Complexity of Counting in Constraint Satisfaction Problems.*

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 396–407, 2010.

when q is non-integer is via the Tutte polynomial, which in its "random cluster" formulation is defined as follows:

$$Z_{\text{Tutte}}(G;q,\gamma) = \sum_{F \subseteq E} q^{\kappa(V,F)} \gamma^{|F|}, \tag{2}$$

where $\kappa(V,F)$ denotes the number of connected components in the graph (V,F). The notation is as before, except that now q is an arbitrary real number. For readers who are familiar with the classical (x,y)-parameterisation of the Tutte polynomial, the transformation between that and the one here is given by $\gamma = y-1$ and $q = (x-1)(y-1)$.

Although (1) and (2) are formally very different, they define the same polynomial in q: see Observation 1. We continue the discussion now in terms of the Tutte polynomial (2), remembering all along that we include as a special case the Potts partition function, and as an even more special case that of the Ising model. We denote by $\textsc{Tutte}(q,\gamma)$ the computational task of computing $Z_{\text{Tutte}}(G;q,\gamma)$ given a graph G as problem instance. Then each pair (q,γ) defines a separate computational problem, and we can study the computational complexity of this problem as q and γ vary. It is important to note that q and γ do not form part of the problem instance, which consists simply of the graph G. For the purposes of this discussion, we may assume that q and γ are rational, in order to avoid representation issues, but in the main body of the paper we work in the wider class of "efficiently approximable" real numbers.

In a seminal paper, Jaeger, Vertigan and Welsh [14] examined the problem of computing $Z_{\text{Tutte}}(G;q,\gamma)$ exactly. In the exact setting, they completely classified the complexity of $\textsc{Tutte}(q,\gamma)$ for all q,γ (in fact for all complex q,γ). It transpires that $\textsc{Tutte}(q,\gamma)$ is #P-hard (i.e., as hard as determining the number of satisfying assignments to a CNF Boolean formula), except when $q=1$, or when (q,γ) is one of a finite number of "special points"; in these cases $\textsc{Tutte}(q,\gamma)$ is polynomial-time computable.

In light of Jaeger et al.'s strong negative result, attention turned to the question of whether $Z_{\text{Tutte}}(G;q,\gamma)$ could be approximated with arbitrarily small specified relative error. In the context of computing partition functions, the appropriate notion of efficient approximate computation is the "Fully polynomial randomised approximation scheme" or FPRAS, which is rigorously defined in §4. An early positive result was provided by Jerrum and Sinclair [15], who presented an FPRAS for the case $q=2$ and $\gamma > 0$, that is to say, for the ferromagnetic Ising model. Sadly, no further generally applicable positive results have appeared since then, though FPRAS's have been proposed for restricted classes of graphs, e.g., dense or degree-bounded [1].

Greater progress has been made in the negative direction. Goldberg and Jerrum [10] showed, under the reasonable complexity-theoretic assumption RP $\neq$ NP, that no FPRAS exists for $\textsc{Tutte}(q,\gamma)$ for a wide range of values for the parameters (q,γ). Stated informally, RP $\neq$ NP is the assumption that there are problems in NP that cannot be decided by a polynomial-time randomised algorithm. As an indicative example of what is known, the intractability result of [10] covers the entire half-plane $\gamma < -2$ except for the tractable case $q=1$

and the case $q = 2$ where the problem is equivalent to approximately counting perfect matchings. Similar results apply when $q/\gamma < -2$. The restriction to planar graphs was treated in a follow-up paper [11]. However none of the existing intractability results apply to the region $q > 0$ and $\gamma > 0$ that concerns us here, and which is perhaps the one of greatest physical interest.

Our goal here is to present the first evidence that $\textsc{Tutte}(q, \gamma)$ is computationally hard in the region $q > 2$ and $\gamma > 0$, i.e., the region corresponding to the ferromagnetic Potts model with $q > 2$ states. We achieve this, but under a stronger complexity-theoretic assumption than RP $\neq$ NP. To explain this assumption, a digression into computational complexity is required.

The complexity class $\#\mathrm{RH}\Pi_1$ of counting problems was introduced by Dyer, Goldberg, Greenhill and Jerrum [6] as a means to classify a wide class of approximate counting problems that were previously of indeterminate computational complexity. The problems in $\#\mathrm{RH}\Pi_1$ are those that can be expressed in terms of counting the number of models of a logical formula from a certain syntactically restricted class. (Although the authors were not aware of it at the time, this syntactically restricted class had already been studied under the title "restricted Krom SNP" [5]. Yet another terminological variation is to say that problems in $\#\mathrm{RH}\Pi_1$ enumerate solutions to a linear Datalog program.) The complexity class $\#\mathrm{RH}\Pi_1$ has a completeness class (with respect to approximation-preserving "AP-reductions") which includes a wide and ever-increasing range of natural counting problems, including: independent sets in a bipartite graph, downsets in a partial order, configurations in the Widom-Rowlinson model (all [6]), the partition function of the ferromagnetic Ising model with mixed external field (i.e., not consistently favouring one or other spin) [9], and stable matchings [4]. Either all of these problems admit an FPRAS (i.e., are efficiently approximable), or none do. No FPRAS is known for any of them at the time of writing, despite much effort having been expended on finding one. All the problems in the completeness class mentioned above are inter-reducible via AP-reductions, so any of them could be said to exemplify the completeness class. However, mainly for historical reasons, the particular problem #BIS, of counting independent sets in a bipartite graph, tends to be taken as the exemplar of the class, much in the same way that Sat has a privileged status in the theory on NP-completeness. Our main result is

Theorem 1. *Suppose that $q > 2$ and $\gamma > 0$ are efficiently approximable. Then* $\#\mathrm{BIS} \leq_{\mathrm{AP}} \textsc{Tutte}(q, \gamma)$.

Here, $\leq_{\mathrm{AP}}$ is the symbol for "is AP-reducible to", and "efficiently approximable" is a concept defined in §4; suffice it to say for now that the rational numbers are trivially efficiently approximable.

One limitation of our inapproximability result is that it is conditional on there being no FPRAS for #BIS (and the rest of the completeness class), rather than on the weaker assumption NP $\neq$ RP. In fact, we conjecture that #BIS does not admit an FPRAS. The basis for our conjecture is empirical — namely that the collection of known #BIS-equivalent problems is growing and that the problem itself has survived its first decade despite considerable efforts to find an

FPRAS. For example, Ge and Štefankovič [7] recently proposed an interesting new MCMC algorithm for sampling indepedent sets in bipartite graphs. Unfortunately, however, the relevant Markov chain mixes slowly in general [8], so even this interesting new idea does not give an FPRAS.

Despite the fact that our results are limited by a strong complexity-theoretic assumption, we feel there are counterbalancing strengths that fully justify this investigation. One is the range and intrinsic interest of the problem under consideration. Whether in the guise of the Potts partition function, or of the Tutte plane, the computational complexity of $\textsc{Tutte}(q, \gamma)$ has received considerable attention since it was first studied by Jaeger et al. [14]: see, for example, [1,21,22,23]. So it seems worth striving for a complexity classification even under a strong assumption such as the one we are making. The situation is similar to working with the Unique Games Conjecture in the area of approximation algorithms for optimisation problems, or employing the class PPAD in analysing the complexity of Nash equilibria. Futhermore, Theorem 1 has a wide range of applicability, covering as it does the whole region $q > 2$, $\gamma > 0$, which, in the classical parameterisation of the Tutte polynomial, equates to the entire upper quadrant of the Tutte plane above the positive branch of the hyperbola $H_2 = \{(x, y) : (x - 1)(y - 1) = 2\}$. Note that the #BIS-hard region extends right to the tractable hyperbola H_2.

Another potential strength of the work is that the reduction introduces a novel technique that may have wider applicability. The idea is conceptually simple and can be sketched informally here. In the first step, we reduce #BIS to a hypergraph version of the Tutte polynomial. This step, if not routine, is at least standard in its techniques. After this, we show how to simulate each hyperedge containing t vertices by a graph gadget with t distinguished vertices or terminals.

At this point we exploit the first order phase transition that is a feature of the so-called random cluster model when $q > 2$. The configurations of the random cluster model on a graph G are spanning subgraphs of G, which are weighted according to the numbers of edges and connected components they contain. As formulated in (2), the Tutte polynomial is the partition function of this model. The gadget is designed and carefully tuned so that is has two coexisting "phases": one in which the random cluster configurations have a large connected (or "giant") component, and one in which they don't. We show that it is possible to arrange for the t terminals to be (with high probability) in a single component in one phase and in t distinct components in the other. This provides us with a bistable gadget that simulates a potentially large hyperedge using many 2-vertex edges. Note that AP-reductions often exploit phase transitions, playing one class of configurations off against another. See the examples in [6] and [17]. What is new here is that, as far as we are aware, this is the first time anyone managed to derive stronger complexity results using the complex phase transitions that arise in actual models studied in statistical physics. Unfortunately, the delicate nature of the gadgets needed to exploit this kind of phase transition does lead to significant technical complexity in our analysis.

Finally, note that Theorem 1 establishes #BIS-hardness of $\textsc{Tutte}(q, \gamma)$ but not #BIS-equivalence. It would be very interesting to know whether there is

an AP-reduction from $\textsc{Tutte}(q,\gamma)$ to #BIS. Note that the complexity of approximate counting is complicated. Bordewich [3] has shown that if any problem in #P fails to have an FPRAS, then there is an infinite approximation hierarchy within #P.

2 Preliminaries

As hinted in §1, we need to generalise the model we are working with. Let $H = (\mathcal{V}, \mathcal{E})$ be a hypergraph with vertex set $\mathcal{V}$ and hyperedge (multi)set $\mathcal{E}$. The multivariate Tutte polynomial of H is defined as follows.

$$Z_{\text{Tutte}}(H; q, \boldsymbol{\gamma}) = \sum_{\mathcal{F} \subseteq \mathcal{E}} q^{\kappa(\mathcal{V},\mathcal{F})} \prod_{f \in \mathcal{F}} \gamma_f,$$

where q and $\boldsymbol{\gamma} = \{\gamma_f\}_{f \in \mathcal{E}}$ are commuting indeterminates and $\kappa(\mathcal{V}, \mathcal{F})$ denotes the number of connected components in the subhypergraph $(\mathcal{V}, \mathcal{F})$. (Two vertices u, v, are in the same component of $(V, \mathcal{F})$ if $u = v$, or there is a sequence $f_1, \ldots, f_\ell \in \mathcal{F}$ of hyperedges with $u \in f_1$, $v \in f_\ell$ and $f_i \cap f_{i+1} \neq \emptyset$ for $1 \leq i < \ell$.) This partition function was studied (under a different name) by Grimmett [12]. An undirected graph G can be viewed as a 2-uniform hypergraph (a hypergraph in which every hyperedge has size 2). In this case, $Z_{\text{Tutte}}(G; q, \boldsymbol{\gamma})$ coincides with the usual definition of the multivariate Tutte polynomial [20].

Let q be a positive integer. The q-state Potts partition function of H is defined as follows:

$$Z_{\text{Potts}}(H; q, \boldsymbol{\gamma}) = \sum_{\sigma : \mathcal{V} \to [q]} \prod_{f \in \mathcal{E}} \big(1 + \gamma_f \delta(\{\sigma(v) \mid v \in f\})\big),$$

where $[q] = \{1, \ldots, q\}$ is a set of q spins or colours, and $\delta(S)$ is 1 if its argument is a singleton and 0 otherwise. The partition function is a sum ranging over assignments of spins to vertices, which are often referred to as "configurations". The following observation is due to Fortuin and Kastelyn.

Observation 1. *If q is a positive integer then $Z_{\text{Potts}}(H; q, \boldsymbol{\gamma}) = Z_{\text{Tutte}}(H; q, \boldsymbol{\gamma})$.*

The *ferromagnetic* Potts model corresponds to the Potts model in the special case in which the edge weights γ_f are non-negative. In this case, a monochromatic edge contributes more weight than an edge with multiple spins. For a subset $\mathcal{F} \subseteq \mathcal{E}$ of the hyperedges of a graph, we use $\gamma(\mathcal{F})$ to denote $\prod_{f \in \mathcal{F}} \gamma_f$.

Consider a graph $G = (V, E)$. Every edge $e \in E$ is associated with a quantity $p(e) \in [0, 1]$. Then for a set of edges $A \subseteq E$ define $\widetilde{P}(G; A, q, p) = q^{\kappa(V,A)} \prod_{e \in A} p(e) \prod_{e \in E \setminus A} (1 - p(e))$. Let

$$Z_{\text{rc}}(G; q, p) = \sum_{A \subseteq E} \widetilde{P}(G; A, q, p) = Z_{\text{Tutte}}(G; q, \boldsymbol{\gamma}) \prod_{e \in E} (1 - p(e)),$$

where $\gamma_e = p(e)/(1 - p(e))$. Then the probability of edge-set A in the random cluster model is given by $P(G; A, q, p) = \widetilde{P}(G; A, q, p)/Z_{\text{rc}}(G; q, p)$. The *random*

cluster model refers to the distribution $\mathrm{RC}(G; q, p)$, in which a subset A of edges is chosen with probability $P(G; A, q, p)$. The difference between Z_{Tutte} and Z_{rc} is simply one of parameterisation. Nevertheless, the change of parameter is useful, as it allows us to employ probabilistic terminology and to exploit existing results from the random graph literature.

3 The Random Cluster Model on Some Natural Graphs

Bollobás, Grimmett and Jansen [2] studied the random cluster model on the complete N-vertex graph K_N. More detailed analyses have since been performed, for example by Luczak and Łuczak [18], but the approach of the earlier paper is easier to adapt to our needs. For fixed q and a fixed constant λ, they studied the distribution $\mathrm{RC}(K_N, q, p)$ where p is the constant function which assigns every edge e of K_N the value $p(e) = \lambda/N$. They show that there is a critical value λ_c, depending on q, so that, if $\lambda > \lambda_c$ then, as $N \to \infty$, with high probability a configuration A drawn from $\mathrm{RC}(K_N, q, p)$ will have a large component (of size linear in N) and otherwise, with high probability the largest component will be much smaller (of size logarithmic in N). For $q > 2$, the critical value is $\lambda_c = 2(q-1)\ln(q-1)/(q-2)$. It is important for our analysis that $\lambda_c < q$ (see [2, p.16]).

Let Γ be the complete graph with vertex set $V_\Gamma = K \cup T$. Let E_Γ denote the edge set of Γ and let $N = |K|$ and $t = |T|$. Let $K^{(2)}$ denote the set of unordered pairs of distinct elements in K and define $T^{(2)}$ similarly. Let ϱ be a value in $[0, 1]$. Define p as follows.

$$p(e) = \begin{cases} \varrho, & \text{if } e \in K^{(2)}, \\ N^{-3/4}, & \text{if } e \in K \times T, \text{ and} \\ 1, & \text{if } e \in T^{(2)}. \end{cases}$$

Ultimately, we will use the graph Γ (or, more precisely, Γ with the edges $T^{(2)}$ deleted) as a gadget to simulate the contribution of a hyperedge on the set T to the multivariate Tutte polynomial of Γ. Thus, we refer to vertices in T as "terminals" of the graph Γ. For a subset $A \subseteq E_\Gamma$, let $Y(A)$ be the number of connected components in the graph $(V_\Gamma, A \setminus T^{(2)})$ that contain terminals. Since, when we come to use the gadget, the edges in $T^{(2)}$ will not be present, we are interested in the structure of connected components in Γ in the absence of these edges, and specifically we are interested in the random variable $Y(A)$. However, it turns out that the key properties of the gadget are easier to verify if we work with a random cluster distribution associated with Γ, exactly as given above, with the edges $T^{(2)}$ present. Informally, the appropriate "boundary condition" for the gadget is the one in which the terminals are joined with probability 1.

The following lemma forms the core of the technical part of the paper. The proof involves comparing the random cluster model to the (multivariate) Erdős-Rényi model of a random graph. This is done using stochastic domination, in the spirit of Holley [13] and also using a multivariate version of Bollobás, Grimmett

and Jansen's "Fundamental Lemma", [2, Lemma 3.1] which studies coloured versions of random cluster configurations in the special case where one of the colour-induced subgraphs is an Erdős-Rényi random graph. See the full version for details.

Lemma 1. *Fix $q > 2$ and let $\lambda = \lambda_c + (q - \lambda_c)/2$. Fix a weight $\gamma > 0$ and let N_0 be a sufficiently large quantity depending on q and γ. Suppose a number of terminals $t > 1$ and a tolerance $0 < \eta < 1$ are given and fix $N \geq \max\{t^{16}, \eta^{-1/8}, N_0\}$. Then there is a parameter ϱ satisfying $N^{-3} \leq \varrho \leq \lambda/N \leq \frac{1}{4}$ such that, if A is drawn from $\mathrm{RC}(\Gamma; q, p)$ then*

$$\Pr(Y(A) = 1) = \gamma \Pr(Y(A) = t). \tag{3}$$

Also, for every value of ϱ in the range $[N^{-3}, \lambda/N]$, if A is drawn from $\mathrm{RC}(\Gamma; q, p)$ then

$$\Pr(1 < Y(A) < t) < \eta. \tag{4}$$

Now let $\widetilde{\Gamma} = (V_\Gamma, E_\Gamma \setminus T^{(2)})$ be the graph derived from Γ by deleting edges within T. Let $\boldsymbol{\gamma} = \{\gamma_e\}$ be the set of edge weights defined by $\gamma_e = p(e)/(1-p(e))$. For an edge subset $A' \subseteq E_\Gamma \setminus T^{(2)}$, let $\kappa'(V_\Gamma, A')$ denote the number of connected components that do not contain terminals in the graph (V_Γ, A'). Let $\mathcal{A}^k$ denote the set of edge subsets $A' \subseteq E_\Gamma \setminus T^{(2)}$ for which the terminals of (V_Γ, A') are contained in exactly k connected components. Let $\mathcal{A} = \bigcup_{k \in [t]} \mathcal{A}^k$ (this is the set of all edge subsets of $\widetilde{\Gamma}$). Let Z^k be q^{-k} times the contribution to $Z_{\mathrm{Tutte}}(\widetilde{\Gamma}; q, \boldsymbol{\gamma})$ from edge subsets $A' \in \mathcal{A}^k$. Formally, $Z^k = \sum_{A' \in \mathcal{A}^k} q^{\kappa'(V, A')} \gamma(A')$. Let $Z = \sum_{k=1}^{t} Z^k$. We will use the following lemma to apply Lemma 1 in our reductions.

Lemma 2. *$Z^k/Z = \Pr(Y(A) = k)$, where A is drawn from $\mathrm{RC}(\Gamma; q, p)$.*

4 Computational Problems, FPRAS's and Efficiently Approximable Real Numbers

Fix real numbers $q > 2$ and $\gamma > 0$ and consider the following computational problem, which is parameterised by q and γ.

Problem. Tutte(q, γ).
Instance. Graph $G = (V, E)$.
Output. $Z_{\mathrm{Tutte}}(G; q, \boldsymbol{\gamma})$, where $\boldsymbol{\gamma}$ is the constant function with $\boldsymbol{\gamma}_e = \gamma$ for every $e \in E$.

We are interested in the complexity of *approximately* solving Tutte(q, γ). We start by defining the relevant concepts. A *randomised approximation scheme* for a function $f : \Sigma^* \to \mathbb{R}$ is a randomised algorithm with the following specification. It takes as input an instance $x \in \Sigma^*$ (e.g., for the problem Tutte(q, γ), the input would be an encoding of a graph G) and a rational error tolerance $\varepsilon > 0$, and outputs a rational number z (a random variable of the "coin tosses" made by the

algorithm) such that, for every instance x, $\Pr\left[e^{-\varepsilon} f(x) \leq z \leq e^{\varepsilon} f(x)\right] \geq \frac{3}{4}$. The randomised approximation scheme is said to be a *fully polynomial randomised approximation scheme*, or *FPRAS*, if it runs in time bounded by a polynomial in $|x|$ and ε^{-1}. The definition is insensitive to the precise success probability, here $3/4$, provided it lies in the interval $(\frac{1}{2}, 1)$ [16, Lemma 6.1]. We say that a real number z is *efficiently approximable* if there is an FPRAS for the problem which maps any input to the output z. Approximations to real numbers are useful. For example, if $\hat{q}$ and $\hat{\gamma}$ are approximations to q and γ satisfying $e^{-\frac{\varepsilon}{n+m}} q \leq \hat{q} \leq e^{\frac{\varepsilon}{n+m}} q$ and $e^{-\frac{\varepsilon}{n+m}} \gamma \leq \hat{\gamma} \leq e^{\frac{\varepsilon}{n+m}} \gamma$ and $\hat{\boldsymbol{\gamma}}_e = \hat{\gamma}$ for every $e \in E$ then

$$e^{-\varepsilon} Z_{\text{Tutte}}(G; q, \boldsymbol{\gamma}) \leq Z_{\text{Tutte}}(G; \hat{q}, \hat{\boldsymbol{\gamma}}) \leq e^{\varepsilon} Z_{\text{Tutte}}(G; q, \boldsymbol{\gamma}).$$

Thus, to approximate $Z_{\text{Tutte}}(G; q, \boldsymbol{\gamma})$, it suffices to first compute rational approximations $\hat{q}$ and $\hat{\gamma}$, and then approximate $Z_{\text{Tutte}}(G; \hat{q}, \hat{\boldsymbol{\gamma}})$.

When the parameters are efficiently approximable reals, it is possible to approximate quantities associated with the gadgets Γ and $\widetilde{\Gamma}$ that we studied in Section 3. We start with the following lemma, which is proved by defining recurrences for the coefficients of the polynomial Z^k. Using the recurrences, the computation can be done by dynamic programming — the fact that q is not known exactly causes no essential problems.

Lemma 3. *Suppose $q > 2$ is an efficiently computable real. Consider the gadget $\widetilde{\Gamma}$ from Section 3 with parameters t, N and ϱ where $\varrho \in [0, 1]$ is a rational number and $N^{1/4}$ is an integer. There is an FPRAS for computing Z^1 and Z^t, given inputs t, N and ϱ.*

It will also be necessary for us to approximate the critical edge probability ϱ, so that, with this approximation, the graph Γ approximately satisfies Equation (3) in Lemma 1. The following lemma shows that this is possible. The algorithm presented in the proof breaks the region $[N^{-3}, \lambda/N]$ into intervals and tries one rational value ϱ within each interval. The computation of $\frac{\Pr(Y(A)=1)}{\Pr(Y(A)=t)}$ uses the algorithm presented in the proof of Lemma 3 (which gives the right answer by Lemma 2) and the fact that the ratio of the probabilities is sufficiently close to γ comes from Lemma 1 and from technical details of the approximation.

Lemma 4. *Suppose $q > 2$ is an efficiently computable real. Fix $\gamma > 0$ and let $\lambda = \lambda_c + (q - \lambda_c)/2$. Suppose that $\chi > 0$ is rational. Consider the gadget Γ from Section 3 with parameters t, N, and ϱ. There is a randomised algorithm whose running time is at most a polynomial in χ^{-1}, N and t which takes input N and t (where it is assumed that $N^{1/4}$ is an integer and that $N \geq \max\{t^{16}, N_0\}$ for the constant N_0 from Lemma 1) and, with probability at least $3/4$, computes a rational ϱ in the range $[N^{-3}, \lambda/N]$ such that, if A is drawn from $\text{RC}(\Gamma; q, p)$, then*

$$e^{-\chi}\gamma \leq \frac{\Pr(Y(A) = 1)}{\Pr(Y(A) = t)} \leq e^{\chi}\gamma.$$

5 Approximation-Preserving Reductions and #BIS

Our main tool for understanding the relative difficulty of approximation counting problems is "approximation-preserving reductions", taken from Dyer, Goldberg, Greenhill and Jerrum [6]. Suppose that f and g are functions from Σ^* to $\mathbb{R}$. An *approximation-preserving reduction* from f to g is a randomised algorithm $\mathcal{A}$ for computing f using an oracle for g. The algorithm $\mathcal{A}$ takes as input a pair $(x, \varepsilon) \in \Sigma^* \times (0, 1)$, and satisfies the following three conditions: (i) every oracle call made by $\mathcal{A}$ is of the form (w, δ), where $w \in \Sigma^*$ is an instance of g, and $0 < \delta < 1$ is an error bound satisfying $\delta^{-1} \leq \text{poly}(|x|, \varepsilon^{-1})$; (ii) the algorithm $\mathcal{A}$ meets the specification for being a randomised approximation scheme for f (as described above) whenever the oracle meets the specification for being a randomised approximation scheme for g; and (iii) the run-time of $\mathcal{A}$ is polynomial in $|x|$ and ε^{-1}. If an approximation-preserving reduction from f to g exists we write $f \leq_{\text{AP}} g$, and say that f *is AP-reducible to* g. Note that if $f \leq_{\text{AP}} g$ and g has an FPRAS then f has an FPRAS. If $f \leq_{\text{AP}} g$ and $g \leq_{\text{AP}} f$ then we say that f *and* g *are AP-interreducible*.

The definitions allow us to construct approximation-preserving reductions between problems f and g with real parameters without insisting that the parameters themselves be efficiently approximable. Nevertheless, some of our results restrict attention to efficiently approximable parameters. According to the definition, approximation-preserving reductions may use randomisation. Nevertheless, the reductions that we present in this paper are deterministic except for where they make use of an FPRAS to approximate a real parameter. A word of warning about terminology: Subsequent to [6], the notation $\leq_{\text{AP}}$ has been used to denote a different type of approximation-preserving reduction which applies to optimisation problems. We will not study optimisation problems in this paper, so hopefully this will not cause confusion.

Dyer et al. [6] studied counting problems in #P and identified three classes of counting problems that are interreducible under approximation-preserving reductions. The first class, containing the problems that admit an FPRAS, are trivially AP-interreducible since all the work can be embedded into the reduction (which declines to use the oracle). The second class is the set of problems that are AP-interreducible with #Sat, the problem of counting satisfying assignments to a Boolean formula in CNF. Zuckerman [24] has shown that #Sat cannot have an FPRAS unless RP = NP. The same is obviously true of any problem to which #Sat is AP-reducible.

The third class appears to be of intermediate complexity. It contains all of the counting problems expressible in the logically-defined complexity class $\#\text{RH}\Pi_1$. Typical complete problems include counting the downsets in a partially ordered set [6], computing the partition function of the ferromagnetic Ising model with varying interaction energies and local external magnetic fields [9] and counting the independent sets in a bipartite graph, which is formally defined as follows.

Problem. #BIS.
Instance. A bipartite graph B.
Output. The number of independent sets in B.

6 The Reductions

This section of the paper gives an approximation-preserving reduction from #BIS to the problem Tutte(q, γ). We start by defining the problem of computing the Tutte polynomial of a uniform hypergraph H with fixed positive edge weights. For fixed positive real numbers q and γ the problem is defined as follows.

Problem. UniformHyperTutte(q, γ)
Instance. A uniform hypergraph $H = (\mathcal{V}, \mathcal{E})$.
Output. $Z_{\text{Tutte}}(H; q, \gamma)$, where $\boldsymbol{\gamma}$ is the constant function with $\boldsymbol{\gamma}_f = \gamma$ for every $f \in \mathcal{E}$.

The following lemma gives an approximation-preserving reduction from #BIS to the problem UniformHyperTutte$(q, q-1)$. The proof goes via an intermediate version of #BIS in which vertices have weights and the degrees of all vertices on the right-hand side are the same.

Lemma 5. *Suppose that $q > 0$ is efficiently approximable. Then* #BIS $\leq_{\text{AP}}$ UniformHyperTutte$(q, q-1)$.

The most difficult part of the paper is reducing UniformHyperTutte(q, γ) to the problem of approximately computing the (multivariate) Tutte polynomial of an undirected graph. Using the random-graphs results from Section 3, we will first (in Lemma 6) show that there is an approximation-preserving reduction to the following intermediate problem.

Problem. TwoWeightFerroTutte(q).
Instance. Graph $G = (V, E)$ with an edge-weight function $\boldsymbol{\gamma}' : E \to \{\gamma', \gamma''\}$ where γ' and γ'' are rationals in the interval $[|V|^{-3}, 1]$.
Output. $Z_{\text{Tutte}}(G; q, \boldsymbol{\gamma}')$.

Lemma 6. *Suppose that $q > 2$ and $\gamma > 0$ are efficiently approximable. Then* UniformHyperTutte$(q, \gamma) \leq_{\text{AP}}$ TwoWeightFerroTutte(q).

The idea behind the proof of Lemma 6 is to use the graph $\widetilde{\Gamma}$ from §3 to simulate a hyperedge. The main technical difficulty lies in showing that the tolerance parameter η controlling inequality (4) in Lemma 1 does not need to be excessively small in order to guarantee sufficient accuracy in this simulation.

The final Lemma 7 completes the chain of AP-reductions from #BIS to Tutte(q, γ) and hence the proof of Theorem 1. The key to this final reduction is a technique for "implementing" an edge of one weight using a subgraph with edges of another weight. The machinery for doing this is taken from [11, Section 1.6]. Suppose G be a graph with edge-weight function $\boldsymbol{\gamma}$, and f is some edge of G with edge weight $\gamma_f = \gamma^*$. We wish to remove the edge of weight γ^* at the expense of introducing several edges of a specified weight γ. We design a graph Υ with distinguished vertices s and t and constant edge weight γ that is equivalent (to a close approximation) to a single edge (s, t) of weight γ^* in any graph context. Consider the weighted graph $\widetilde{G}$ obtained by removing edge f and

replacing it with a copy of Υ, identifying s and t with the endpoints of f). We require of Υ the property that the Tutte polynomial of $\widetilde{G}$ is close to the Tutte polynomial of G, up to an easily computable factor. Using this idea, following the pattern of [11], but paying particular attention to technical issues arising because of approximation of real numbers, establishes the last link in the chain of reductions.

Lemma 7. *Suppose that $q > 2$ and $\gamma > 0$ are efficiently approximable. Then* TwoWeightFerroTutte$(q) \leq_{\mathrm{AP}}$ Tutte(q, γ).

7 3-Uniform Hypergraphs

Lemma 5 has the following corollary.

Corollary 1. #BIS $\leq_{\mathrm{AP}}$ UniformHyperTutte$(q, q-1)$ *for efficiently approximable $q > 0$.*

Thus, assuming that there is no FPRAS for #BIS, we can conclude that there is no FPRAS for computing the Tutte polynomial of a uniform hypergraph when the edge-weights are set to $\gamma = q - 1$. The *Ising model* corresponds to the $q = 2$ case of the Potts model. Thus, we conclude that there is no FPRAS for computing the partition function of the Ising model on a uniform hypergraph in which every edge has weight 1. We conclude this paper with a contrasting positive result for 3-uniform hypergraphs. Consider the following problem.

Problem. 3-UniformHyperTutte(q, γ).
Instance. A 3-uniform hypergraph $H = (\mathcal{V}, \mathcal{E})$.
Output. $Z_{\mathrm{Tutte}}(H; q, \gamma)$, where $\boldsymbol{\gamma}$ is the constant function with $\boldsymbol{\gamma}_f = \gamma$ for every $f \in \mathcal{E}$.

Lemma 8. *Suppose that $\gamma > 0$ is efficiently approximable. There is an FPRAS for the problem* 3-UniformHyperTutte$(2, \gamma)$.

References

1. Alon, N., Frieze, A., Welsh, D.: Polynomial time randomized approximation schemes for Tutte-Gröthendieck invariants: the dense case. Random Structures Algorithms 6(4), 459–478 (1995)
2. Bollobás, B., Grimmett, G., Janson, S.: The random-cluster model on the complete graph. Probab. Theory Related Fields 104(3), 283–317 (1996)
3. Bordewich, M.: On the approximation complexity hierarchy (in preparation, 2010)
4. Chebolu, P., Goldberg, L.A., Martin, R.: Approximately counting stable matchings (in preparation, 2010)
5. Dalmau, V.: Linear datalog and bounded path duality of relational structures. Logical Methods in Computer Science 1(1) (2005)
6. Dyer, M.E., Goldberg, L.A., Greenhill, C.S., Jerrum, M.: The relative complexity of approximate counting problems. Algorithmica 38(3), 471–500 (2003)

7. Ge, Q., Stefankovic, D.: A graph polynomial for independent sets of bipartite graphs. CoRR, abs/0911.4732 (2009)
8. Goldberg, L.A., Jerrum, M.: Counterexample to rapid mixing of the GS Process. Technical note (2010)
9. Goldberg, L.A., Jerrum, M.: The complexity of ferromagnetic Ising with local fields. Combinatorics, Probability & Computing 16(1), 43–61 (2007)
10. Goldberg, L.A., Jerrum, M.: Inapproximability of the Tutte polynomial. Inform. and Comput. 206(7), 908–929 (2008)
11. Goldberg, L.A., Jerrum, M.: Inapproximability of the Tutte polynomial of a planar graph. CoRR, abs/0907.1724 (2009)
12. Grimmett, G.: Potts models and random-cluster processes with many-body interactions. J. Statist. Phys. 75(1-2), 67–121 (1994)
13. Holley, R.: Remarks on the FKG inequalities. Comm. Math. Phys. 36, 227–231 (1974)
14. Jaeger, F., Vertigan, D.L., Welsh, D.J.A.: On the computational complexity of the Jones and Tutte polynomials. Math. Proc. Cambridge Philos. Soc. 108(1), 35–53 (1990)
15. Jerrum, M., Sinclair, A.: Polynomial-time approximation algorithms for the Ising model. SIAM J. Comput. 22(5), 1087–1116 (1993)
16. Jerrum, M.R., Valiant, L.G., Vazirani, V.V.: Random generation of combinatorial structures from a uniform distribution. Theoret. Comput. Sci. 43(2-3), 169–188 (1986)
17. Kelk, S.: On the relative complexity of approximately counting H-colourings. PhD thesis, University of Warwick, Coventry, UK (July 2004)
18. Luczak, M., Łuczak, T.: The phase transition in the cluster-scaled model of a random graph. Random Structures Algorithms 28(2), 215–246 (2006)
19. Potts, R.B.: Some generalized order-disorder transformations. Proc. Cambridge Philos. Soc. 48, 106–109 (1952)
20. Sokal, A.: The multivariate Tutte polynomial. In: Surveys in Combinatorics. Cambridge University Press, Cambridge (2005)
21. Vertigan, D.L., Welsh, D.J.A.: The computational complexity of the Tutte plane: the bipartite case. Combin. Probab. Comput. 1(2), 181–187 (1992)
22. Vertigan, D.: The computational complexity of Tutte invariants for planar graphs. SIAM J. Comput. 35(3), 690–712 (2005) (electronic)
23. Welsh, D.J.A.: Complexity: knots, colourings and counting. London Mathematical Society Lecture Note Series, vol. 186. Cambridge University Press, Cambridge (1993)
24. Zuckerman, D.: On unapproximable versions of NP-Complete problems. SIAM Journal on Computing 25(6), 1293–1304 (1996)

On the Relation between Polynomial Identity Testing and Finding Variable Disjoint Factors

Amir Shpilka and Ilya Volkovich

Faculty of Computer Science, Technion, Haifa 32000, Israel
{shpilka,ilyav}@cs.technion.ac.il.

Abstract. We say that a polynomial $f(x_1, \ldots, x_n)$ is *indecomposable* if it cannot be written as a product of two polynomials that are defined over disjoint sets of variables. The *polynomial decomposition* problem is defined to be the task of finding the indecomposable factors of a given polynomial. Note that for multilinear polynomials, factorization is the same as decomposition, as any two different factors are variable disjoint.

In this paper we show that the problem of derandomizing polynomial identity testing is essentially equivalent to the problem of derandomizing algorithms for polynomial decomposition. More accurately, we show that for any reasonable circuit class there is a deterministic polynomial time (black-box) algorithm for polynomial identity testing of that class if and only if there is a deterministic polynomial time (black-box) algorithm for factoring a polynomial, computed in the class, to its indecomposable components.

An immediate corollary is that polynomial identity testing and polynomial factorization are equivalent (up to a polynomial overhead) for multilinear polynomials. In addition, we observe that derandomizing the polynomial decomposition problem is equivalent, in the sense of Kabanets and Impagliazzo [1], to proving arithmetic circuit lower bounds for NEXP.

Our approach uses ideas from [2], that showed that the polynomial identity testing problem for a circuit class $\mathcal{C}$ is essentially equivalent to the problem of deciding whether a circuit from $\mathcal{C}$ computes a polynomial that has a read-once arithmetic formula.

1 Introduction

In this paper we study the relation between two fundamental algebraic problems, polynomial identity testing and polynomial factorization. We show that the tasks of giving deterministic algorithms for polynomial identity testing and for a variant of the factorization problem (that we refer to as the polynomial decomposition problem) are essentially equivalent. We first give some background on both problems and then discuss our results in detail.

Polynomial Decomposition. Let $X = (x_1, \ldots, x_n)$ be the set of variables. For a set $I \subseteq [n]$ denote with X_I the set of variables whose indices belong to I. A polynomial f, depending on X, is said to be *decomposable* if it can be written

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 408–419, 2010.

as $f(X) = g(X_S) \cdot h(X_{[n]\setminus S})$ for some $\emptyset \subsetneq S \subsetneq [n]$. The *indecomposable factors* of a polynomial $f(X)$ are polynomials $f_1(X_{I_1}), \ldots, f_k(X_{I_k})$ such that the I_j-s are disjoint sets of indices, $f(X) = f_1(X_{I_1}) \cdot f_2(X_{I_2}) \cdots f_k(X_{I_k})$ and the f_i's are indecomposable. It is not difficult to see that every polynomial has a unique factorization to indecomposable factors (up to multiplication by field elements). The problem of polynomial decomposition is defined in the following way: Given an arithmetic circuit from an arithmetic circuit class $\mathcal{C}$ computing a polynomial f, we have to output circuits for each of the indecomposable factors of f. If we only have a black-box access to f then we have to output a black-box (i.e. an algorithm that may use the original black-box) for each of the indecomposable factors of f. Clearly, finding the indecomposable factors of a polynomial f is an easier task than finding all the irreducible factors of f. It is not hard to see though, that for the natural class of multilinear polynomials the two problems are the same. We also consider the decision version of the polynomial decomposition problem: Given an arithmetic circuit computing a multivariate polynomial decide whether the polynomial is decomposable or not. Note that in the decision version the algorithm just has to answer 'yes' or 'no' and is not required to find the decomposition.

Many randomized algorithms are known for factoring multivariate polynomials in the black-box and non black-box models (see the surveys in [3,4,5]). These algorithms also solve the decomposition problem. However, it is a long standing open question whether there is an efficient *deterministic* algorithm for factoring multivariate polynomials (see [3,6]). Moreover, there is no known deterministic algorithm even for the decision version of the problem (that is defined analogously). Furthermore, even for the simpler case of factoring multilinear polynomials (which is a subproblem of polynomial decomposition) no deterministic algorithms are known.

Polynomial Identity Testing. Let $\mathcal{C}$ be a class of arithmetic circuits defined over some field $\mathbb{F}$. The polynomial identity testing problem (PIT for short) for $\mathcal{C}$ is the question of deciding whether a given circuit form $\mathcal{C}$ computes the identically zero polynomial. This question can be considered both in the black-box model, in which we can only access the polynomial computed by the circuit using queries, or in the non black-box model where the circuit is given to us. The importance of this fundamental problem stems from its many applications. For example, the deterministic primality testing algorithm of [7] and the fast parallel algorithm for perfect matching of [8] are based on solving PIT problems.

PIT has a well known randomized algorithm [9,10,11]. However, we are interested in the problem of obtaining efficient *deterministic* algorithms for it. This question received a lot of attention recently [12,13,14,1,15,16,17,18,19,20,21,2,22,23,24,25,26,27,28] but its deterministic complexity is still far from being well understood. In [29,1,15,22] results connecting PIT to lower bounds for arithmetic circuits where proved, shedding light on the difficulty of the problem. It is interesting to note that the PIT problem becomes very difficult already for depth-4 circuits. Indeed, [23] proved that a polynomial time black-box PIT

algorithm for depth-4 circuits implies an exponential lower bound for general arithmetic circuits (and hence using the ideas of [1] a quasi-polynomial time deterministic PIT algorithm for general circuits).

In this work we (essentially) show equivalence between the PIT and polynomial decomposition problems. Namely, we prove that for any (reasonable) circuit class $\mathcal{C}$, it holds that $\mathcal{C}$ has a polynomial time deterministic PIT algorithm if and only if it has a polynomial time deterministic decomposition algorithm. The result holds both in the black-box and the non black-box models. That is, if the PIT for $\mathcal{C}$ is in the black-box model then deterministic black-box decomposition is possible and vice versa, and similarly for the non black-box case. Before giving the formal statement of our results we give some definitions.

Arithmetic Circuits. An *arithmetic circuit* in the variables $X = \{x_1, \ldots, x_n\}$, over the field $\mathbb{F}$, is a labelled directed acyclic graph. The inputs (nodes of in-degree zero) are labelled by variables from X or by constants from the field. The internal nodes are labelled by $+$ or $\times$, computing the sum and product, resp., of the polynomials on the tails of incoming edges (subtraction is obtained using the constant -1). A *formula* is a circuit whose nodes have out-degree one (namely, a tree). The output of a circuit (formula) is the polynomial computed at the output node. The *size* of a circuit (formula) is the number of gates in it. The *depth* of the circuit (formula) is the length of a longest path between the output node and an input node.

We shall say that a polynomial $f(x_1, \ldots, x_n)$ has *individual degrees* bounded by d, if no variable has degree higher than d in f. An arithmetic circuit C has *individual degrees* bounded by d if the polynomial that C computes has individual degrees bounded by d. Finally, we shall say that C is an (n, s, d)-*circuit* if it is an n-variate arithmetic circuit of size s with individual degrees bounded by d. Sometimes we shall think of an arithmetic circuit and of the polynomial that it computes as the same objects.

In this paper we will usually refer to a model of arithmetic circuits $\mathcal{C}$. It should be thought of as either the general model of arithmetic circuits or as some restricted model such as bounded depth circuits, etc.

Our Results. We now formally state our results. We give them in a very general form as we later apply them to very restricted classes such as depth-3 circuits, read-once formulas etc.

Theorem 1 (Main). *Let $\mathcal{C}$ be a class of arithmetic circuits, defined over a field $\mathbb{F}$. Consider circuits of the form $C = C_1 + C_2 \times C_3$, where the C_i-s are (n, s, d)-circuits from $\mathcal{C}$ and, C_2 and C_3 are defined over disjoint sets of variables.*[1] *Assume that there is a deterministic algorithm that when given access (explicit or via a black-box) to such a circuit C runs in time $T(s, d)$ and decides whether $C \equiv 0$. Then, there is a deterministic algorithm that when given*

[1] This requirement seems a bit strange but we need it in order to state our results in the most general terms.

access (explicit or via a black-box) to an (n, s, d)-circuit $C' \in \mathcal{C}$,[2] *runs in time $\mathcal{O}(n^3 \cdot d \cdot T(s, d))$ and outputs the indecomposable factors, $H = \{h_1, \ldots, h_k\}$, of the polynomial computed by C'. Moreover, each h_i is in $\mathcal{C}$ and $size(h_i) \leq s$.*

The other direction is, in fact, very easy and is given by the following observation.

Observation 1. *Let $\mathcal{C}$ be a class of arithmetic circuits. Assume that there is an algorithm that when given access (explicit or via a black-box) to an (n, s, d)-circuit $C \in \mathcal{C}$ runs in time $T(s, d)$ and outputs "true" iff the polynomial computed by C is decomposable. Then, there is a deterministic algorithm that runs in time $\mathcal{O}(T(s + 2, d))$ and solves the PIT problem for size s circuits from $\mathcal{C}$.*

As mentioned above, the irreducible factors of multilinear polynomials are simply their indecomposable factors. Hence we obtain the following corollary. We give here a slightly informal statement. The full statement is given in Section 3.1.

Corollary 1 (informal). *Let $\mathcal{C}$ be an arithmetic circuit class computing multilinear polynomials. Then, up to a polynomial overhead, the deterministic polynomial identity testing problem and the deterministic factorization problem for circuits from $\mathcal{C}$ are equivalent, in both the black-box the and non black-box models.*

We also obtain some extensions to the results above. The first result shows how to get a non-adaptive decomposition from a PIT algorithm (Theorem 1 gives an adaptive algorithm). To prove it we need a stronger PIT algorithm than the one used in the proof of Theorem 1. The second extension gives an algorithm deciding whether for a given polynomial f there are two variables x_i, x_j such that $f(X) = f_1(X_{[n]\setminus\{i\}}) \cdot f_2(X_{[n]\setminus\{j\}})$. This can be thought of as a generalization of Theorem 1. Finally, we obtain a connection between the decomposition problem and lower bounds in the sense of Kabanets and Impagliazzo. Due to space limitations the proofs of those results are omitted from this version.

Motivation. The motivation for this work is twofold. First, the most obvious motivation is that we think that the problem of connecting the complexity of PIT and polynomial factorization is very natural. Another motivation is to better understand the PIT problem for multilinear formulas.[3] Although lower bounds are known for multilinear formulas [30, 31, 32, 33], we do not have an efficient PIT algorithm even for depth-3 multilinear formulas. Consider the following approach towards PIT of multilinear formulas. Start by making the formula a read-once formula. I.e. a formula in which every variable labels at most one leaf. This can be done by replacing, for each i and j, the j-th occurrence of x_i with a new variable $x_{i,j}$. Now, using PIT algorithm for read-once formulas [2, 26], check whether this formula is zero or not. If it is zero then the original formulas was also zero and we are done. Otherwise start replacing back each $x_{i,j}$ with x_i. After each replacement we would like to verify

[2] $C' \in \mathcal{C}$ denotes that the circuit C' is from $\mathcal{C}$.

[3] A multilinear formula is a formula in which every gate computes a multilinear polynomial, see [30].

that the resulting formula is still not zero. Notice that when replacing $x_{i,j}$ with x_i we get zero if and only if the linear function $x_i - x_{i,j}$ is a factor of the formula at hand. Thus, we somehow have to find a way of verifying whether a linear function is a factor of a multilinear formula. Notice that as we start with a read-once formula for which PIT is known [2,26], we can assume that we know many inputs on which the formula does not vanish. One may hope that before replacing $x_{i,j}$ with x_i we somehow managed to obtain inputs that will enable us to verify whether $x_i - x_{i,j}$ is a factor of the formula or not. This of course is not formal and only gives a sketch of an idea, but it shows the importance of understanding how to factor multilinear formulas given a PIT algorithm. As different factors of multilinear formulas are variable disjoint this motivates the study of polynomial decomposition.

Proof Technique. It is not difficult to see that efficient algorithms for polynomial decomposition imply efficient algorithms for PIT. Indeed, notice that $f(x_1, \ldots, x_n) \equiv 0$ iff $f(x_1, \ldots, x_n) + y \cdot z$, where y and z are two new variables, is decomposable (in which case y and z are its indecomposable factors). Hence, an algorithm for polynomial decomposition (even for the decision version of the problem) gives rise to a PIT algorithm.

The more interesting direction is obtaining a decomposition algorithm given a PIT algorithm. Note that if $f(X) = f_1(X_{I_1}) \cdot \ldots \cdot f_k(X_{I_k})$ is the decomposition of f and if we know the sets $I_1, \ldots, I_k$ then using the PIT algorithm we can easily obtain circuits for the different f_i's. Indeed, if $\bar{a} \in \mathbb{F}^n$ is such that $f(\bar{a}) \neq 0$ then, for some constant α_j, $f_j(X_{I_j}) = \alpha_j \cdot f|_{\bar{a}_{[n]\setminus I_j}}(X)$, where $\bar{a}_{[n]\setminus I_j}$ is the assignment that assigns values to all the variables except those whose index belongs to I_j.[4] Now, given a PIT algorithm we can use it to obtain such $\bar{a}$ in a manner similar to finding a satisfying assignment to a CNF formula given a SAT oracle. Consequently, finding the partition $I_1, \ldots, I_k$ of $[n]$ is equivalent to finding the indecomposable factors (assuming that we have a PIT algorithm).

We present two approaches for finding the partition. The first is by induction: Using the PIT algorithm we obtain an assignment $\bar{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$ that has the property that for every $j \in [n]$ it holds that f depends on x_j if and only if $f|_{\bar{a}_{[n]\setminus\{j\}}}$ depends on x_j. Following [34,35,2,26] we call $\bar{a}$ a *justifying* assignment of f. Given a justifying assignment $\bar{a}$, we find, by induction, the indecomposable factors of $f|_{x_n=a_n}$. Then, using simple algebra we recover the indecomposable factors of f from those of $f|_{x_n=a_n}$. This is the idea behind the proof of Theorem 1.

In the second approach, we observe that the variables x_i and x_j belong to the same set I in the partition iff $\Delta_{ij} f \stackrel{\Delta}{=} f \cdot f|_{x_i=y,x_j=w} - f|_{x_i=y} \cdot f|_{x_j=w} \not\equiv 0$, when y and w are two new variables. Using this observation we obtain the partition by constructing a graph G on the set of vertices $[n]$ in which i and j are neighbors if and only if $\Delta_{ij} f \not\equiv 0$. The sets of the partition are exactly the connected components of G. Due to space limitations we present only the first approach.

[4] In fact, we also need a constant α, that is easily computable, to get that $f(X) = \alpha \cdot f_1(X_{I_1}) \cdot \ldots \cdot f_k(X_{I_k})$.

Related Works. The only work that we are aware of that studies the relation between PIT and polynomial factorization is [36]. There, Kaltofen and Koiran gave polynomial time deterministic identity testing algorithm for a class of super-sparse univariate polynomials, and from there obtained a deterministic factorization algorithm for a related class of bivariate polynomials. This model is, of course, very different from the models studied here.

As mentioned above, justifying assignments were first defined and used in [34,35] for the purpose of giving randomized polynomial time learning algorithms for read-once arithmetic formulas. In [2,26] justifying assignments were used in conjunction with new PIT algorithms in order to obtain deterministic quasi-polynomial time interpolation algorithms for read-once formulas. We rely on the ideas from [26] for obtaining justifying assignments from PIT algorithms.

Another line of works that is related to our results is that of Kabanets and Impagliazzo [1] and of [22]. There it was shown that the question of derandomizing PIT is closely related to the problem of proving lower bounds for arithmetic circuits. These results use the fact that factors of small arithmetic circuits can also be computed by small arithmetic circuits. This gives another connection between PIT and polynomial factorization, although a less direct one.

The results of [1] relate PIT to arithmetic lower bounds for NEXP. However, these lower bounds are not strong enough and do not imply that derandomization of PIT gives derandomization of other algebraic problems. Similarly, the results of [15] show that polynomial time *black-box* PIT algorithms give rise to exponential lower bounds for arithmetic circuits which in turn, using ideas a-la [1], may give *quasi-polynomial* time derandomization of polynomial factorization.[5] However, this still does not guarantee polynomial time derandomization as is achieved in this work.

Organization. In Section 2 we give the required definition and discuss partial derivatives and justifying assignments. In Section 3 we prove our main result and derive some corollaries.

2 Preliminaries

For an integer n denote $[n] = \{1, \ldots, n\}$. In this paper all the circuits and polynomials are defined over some field $\mathbb{F}$. In most of our algorithms we will need to assume that $\mathbb{F}$ is larger than some function depending on n (we will mostly have the requirement $|\mathbb{F}| > nd$, where n is the number of variables and d is an upper bound on the individual degrees of the given circuit/polynomial). We note that this is not a real issue as in most works on factoring or on PIT it is assumed that we can access a polynomially large extension field of $\mathbb{F}$. From now on we assume that $\mathbb{F}$ is sufficiently large.

For a polynomial $f(x_1, \ldots, x_n)$, a variable x_i and a field element α we denote with $f|_{x_i=\alpha}$ the polynomial resulting from substituting α to x_i.

[5] We note that, currently, it is not clear how to derandomize the factorization problem using lower bounds for arithmetic circuits.

Similarly, given a subset $I \subseteq [n]$ and an assignment $\bar{a} \in \mathbb{F}^n$ we define $f|_{x_I=\bar{a}_I}$ to be the polynomial resulting from substituting a_i to x_i for every $i \in I$. We say that f *depends* on x_i if there exist $\bar{a} \in \mathbb{F}^n$ and $b \in \mathbb{F}$ such that: $f(a_1, a_2, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n) \neq f(a_1, a_2, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_n)$. We denote $\mathrm{var}(f) \stackrel{\Delta}{=} \{i \in [n] \mid f \text{ depends on } x_i\}$. It is not difficult to see that f depends on x_i if and only if x_i appears when f is written as a sum of monomials. By substituting a value to a variable of f we obviously eliminate the dependence of f on this variable. However, this can also eliminate the dependence of f on other variables, so we may lose more 'information' than intended. To handle this problem we define a 'lossless' type of an assignment. Similar definitions were given in [34,35,2,26]. For completeness we repeat the definitions here.

Definition 1 (Justifying assignment). *Given an assignment $\bar{a} \in \mathbb{F}^n$ we say that $\bar{a}$ is a* justifying assignment *of f if for every subset $I \subseteq \mathrm{var}(f)$ we have that $\mathrm{var}(f|_{x_I=\bar{a}_I}) = \mathrm{var}(f) \setminus I$.*

Proposition 1 ([2]). *An assignment $\bar{a} \in \mathbb{F}^n$ is a justifying assignment of f if and only if $\mathrm{var}(f|_{x_I=\bar{a}_I}) = \mathrm{var}(f) \setminus I$ for every subset I of size $|I| = |\mathrm{var}(f)| - 1$.*

We now show how to get a justifying assignment from a polynomial identity testing algorithm. This was first done in [2] (and generalized in [26]). Before stating the result we shall need the following definition.

Definition 2 (Partial Derivative). *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. The* partial derivative *of f w.r.t. x_i and direction $\alpha \in \mathbb{F}$ is defined as $\frac{\partial f}{\partial_\alpha x_i} \stackrel{\Delta}{=} f|_{x_i=\alpha} - f|_{x_i=0}$. For an arithmetic circuit C we define $\frac{\partial C}{\partial_\alpha x_i} \stackrel{\Delta}{=} C|_{x_i=\alpha} - C|_{x_i=0}$.*

Our algorithm will consider a circuit class $\mathcal{C}$ but will require a PIT algorithm for a slightly larger class. Namely, for every circuit $C \in \mathcal{C}$ we will need a PIT algorithm for circuits of the form $\frac{\partial C}{\partial_\alpha x_i}$. To ease the reading we shall refer to all circuits of the form $\frac{\partial C}{\partial_\alpha x_i}$ as ∂C.

Theorem 2 ([2,26]). *Let $\mathbb{F}$ be a field of size $|\mathbb{F}| \geq nd$. Let f be a polynomial that is computed by an (n, s, d)-circuit $C \in \mathcal{C}$. Then, there is an algorithm that returns a justifying assignment $\bar{a}$ for f in time $\mathcal{O}(n^3 d \cdot T(s, d))$, where $T(s, d)$ is the running time of the PIT algorithm for circuits of the form ∂C where $C \in \mathcal{C}$ is an (n, s, d)-circuit.*

2.1 Indecomposable Polynomials

Definition 3. *We say that a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is* indecomposable *if it is non-constant and cannot be represented as the product of two (or more) non-constant variable disjoint polynomials. Otherwise, we say that f is decomposable.*

Clearly decomposability is a relaxation of irreducibility. For example, $(x + y + 1)(x + y - 1)$ is indecomposable but is not irreducible. Also note that any univariate polynomial is indecomposable. The following lemma is easy to prove.

Lemma 1 (Unique decomposition). *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-constant polynomial. Then f has a unique (up to multiplication by field elements) factorization to indecomposable factors.*

Observation 2. *Let f be a multilinear polynomial. Then f is indecomposable if and only if f is irreducible. In particular, if $f(\bar{x}) = f_1(\bar{x}) \cdot f_2(\bar{x}) \cdot \ldots \cdot f_k(\bar{x})$ is the decomposition of f, then the f_i-s are f's irreducible factors.*

3 Decomposition

In this section we give the proof of Theorem 1. Algorithm 1 shows how to find the indecomposable factors for a polynomial computed by $\mathcal{C}$ using the PIT algorithm. In fact, the algorithm returns a partition $\mathcal{I} = \{I_1, \ldots, I_k\}$ of $[n]$ such that the decomposition of f is $f = h_1(X_{I_1}) \cdot \ldots \cdot h_k(X_{I_k})$, for some polynomials $h_1, \ldots, h_k$. We call $\mathcal{I}$ the *variable-partition* of f. The idea behind the algorithm is to first find a justifying assignment $\bar{a}$ to f using Theorem 2. Then, to find the partition of $f|_{x_n=a_n}$. Finally, by using the PIT algorithm, to decide which sets in the partition of $f|_{x_n=a_n}$ belong to the partition of f and which sets must be unified.

Algorithm 1. Finding variable partition

Input: An (n, s, d)-circuit C from a circuit class $\mathcal{C}$, a justifying assignment $\bar{a}$ for C, and access to a PIT algorithm as in the statement of Theorem 1.
Output: A variable-partition $\mathcal{I}$

1: Set $\mathcal{I} = \emptyset$, $J = [n]$ ($\mathcal{I}$ will be the partition that we seek).
2: Set $x_n = a_n$ and recursively compute the variable-partition of $C' = C|_{x_n=a_n}$. Let $\mathcal{I}'$ be the resulting partition (note that when $n = 1$ then we just return $\mathcal{I} = \{\{1\}\}$).
3: For every set $I \in \mathcal{I}'$ check whether $C(\bar{a}) \cdot C \equiv C|_{x_I=\bar{a}_I} \cdot C|_{x_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$. If this is the case then add I to $\mathcal{I}$ and set $J \leftarrow J \setminus I$. Otherwise, move to the next I.
4: Finally, add the remaining elements to $\mathcal{I}$. Namely, $\mathcal{I} \leftarrow \mathcal{I} \cup \{J\}$.

The following lemma gives the analysis of the algorithm and its correctness.

Lemma 2. *Let C be an (n, s, d)-circuit from $\mathcal{C}$ such that $\mathrm{var}(C) = [n]$. Assume there exists a PIT algorithm as in the statement of Theorem 1. Let $\bar{a}$ be a justifying assignment of C. Then given C and $\bar{a}$ Algorithm 1 outputs a variable-partition $\mathcal{I}$ for the polynomial computed by C. The running time of the algorithm is $\mathcal{O}(n^2 \cdot T(s, d))$, where $T(s, d)$ is as in the statement of Theorem 1.*

Proof. The proof of correctness is by induction on n. For the base case ($n = 1$) we recall that a univariate polynomial is an indecomposable polynomial. Now assume that $n > 1$. Let $C = h_1(X_{I_1}) \cdot \ldots \cdot h_{k-1}(X_{I_{k-1}}) \cdot h_k(X_{I_k})$ be the decomposition of C where $\mathcal{I} = \{I_1, \ldots, I_k\}$ is its variable-partition. Assume w.l.o.g. that $n \in I_k$. Consider $C' = C|_{x_n=a_n}$. It holds that $C' = C|_{x_n=a_n} = h_1 \cdot \ldots \cdot h_{k-1} \cdot h_k|_{x_n=a_n} = h_1 \cdot \ldots \cdot h_{k-1} \cdot g_1 \cdot g_2 \cdot \ldots \cdot g_\ell$ where the g_i-s are

the indecomposable factors of $h_k|_{x_n=a_n}$. Denote with $\mathcal{I}_k = \{I_{k,1}, \ldots, I_{k,\ell}\}$ the variable-partition of $h_k|_{x_n=a_n}$. As $\bar{a}$ is a justifying assignment of C we obtain that $\mathrm{var}(C') = [n-1]$. From the uniqueness of the decomposition (Lemma 1) and by the induction hypothesis we get that, when running on C', the algorithm returns $\mathcal{I}' = \{I_1, \ldots, I_{k-1}, I_{k,1}, \ldots, I_{k,\ell}\}$. The next lemma shows that the algorithm indeed returns the variable-partition $\mathcal{I}$.

Lemma 3. *Let $f(\bar{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial and let $\bar{a} \in \mathbb{F}^n$ be a justifying assignment of f. Then $I \subseteq [n]$ satisfies that $f(\bar{a}) \cdot f \equiv f|_{x_I=\bar{a}_I} \cdot f|_{x_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$, if and only if I is a disjoint union of sets from the variable-partition of f.*

Proof. Assume that equality holds. Then, as $\bar{a}$ is a justifying assignment of f we have that $f|_{x_I=\bar{a}_I}, f|_{x_{[n]\setminus I}=\bar{a}_{[n]\setminus I}} \not\equiv 0$ and hence $f(\bar{a}) \neq 0$. Consequently, if we define $h(X_I) \stackrel{\Delta}{=} f|_{x_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$ and $g(X_{[n]\setminus I}) \stackrel{\Delta}{=} \frac{f|_{x_I=\bar{a}_I}}{f(\bar{a})}$ then we obtain that $f(\bar{x}) = h(X_I) \cdot g(X_{[n]\setminus I})$. The result follows by uniqueness of decomposition.

To prove the other direction notice that we can write $f(\bar{x}) \equiv h(X_I) \cdot g(X_{[n]\setminus I})$ for two polynomials h and g. We now have that, $f|_{x_I=\bar{a}_I} \equiv h(\bar{a}) \cdot g(X_{[n]\setminus I})$ and similarly $f|_{x_{[n]\setminus I}=\bar{a}_{[n]\setminus I}} \equiv h(X_I) \cdot g(\bar{a})$. Hence, $f(\bar{a}) \cdot f \equiv h(\bar{a}) \cdot g(\bar{a}) \cdot h(X_I) \cdot g(X_{[n]\setminus I}) \equiv f|_{x_I=\bar{a}_I} \cdot f|_{x_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$. This concludes the proof of Lemma 3. □

By the lemma, each I_j $(j < k)$ will be added to $\mathcal{I}$ whereas no $I_{k,j}$ will be added to it. Eventually we will have that $J = I_k$ as required. To finish the proof of Lemma 2 we now analyze the running time of the algorithm. The following recursion is satisfied, where $t(n,s,d)$ is the running time of the algorithm on input an (n,s,d)-circuit $C \in \mathcal{C}$: $t(n,s,d) = t(n-1,s,d) + \mathcal{O}(|\mathcal{I}'| \cdot T(s,d)) = t(n-1,s,d) + \mathcal{O}(n \cdot T(s,d))$, which implies that $t(n,s,d) = \mathcal{O}(n^2 \cdot T(s,d))$. □

The proof of Theorem 1 easily follows.

Proof (of Theorem 1). We first note that the assumed PIT algorithm also works for circuits in ∂C, when C is an (n,s,d)-circuit from $\mathcal{C}$. Therefore, by Theorem 2 we have an algorithm that finds a justifying assignment $\bar{a}$, as well as computes $\mathrm{var}(C)$.[6] This requires $\mathcal{O}(n^3 \cdot d \cdot T(s,d))$ time. Once $\mathrm{var}(C)$ is known we can assume w.l.o.g that $\mathrm{var}(C) = [n]$. Lemma 2 guarantees that Algorithm 1 returns a variable-partition $\mathcal{I}$ in time $\mathcal{O}(n^2 \cdot T(s,d))$. At this point we can define, for every $I \in \mathcal{I}$ the polynomial $h_I \stackrel{\Delta}{=} C|_{x_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$. It is now clear that for $\alpha = C(\bar{a})^{1-|\mathcal{I}|}$ we have that $C = \alpha \prod_{I \in \mathcal{I}} h_I$ is the decomposition of C. Moreover, note that from the definition, each h_i belongs to $\mathcal{C}$ and has size at most s. The total running time can be bounded from above by $\mathcal{O}(n^3 \cdot d \cdot T(s,d))$. □

To complete the equivalence between polynomial decomposition and PIT we provide a short proof of Observation 1.

Proof (of Observation 1). Let C be an arithmetic circuit. Consider $C' \stackrel{\Delta}{=} C + y \cdot z$ where y, z are new variables. Clearly, C' is decomposable iff $C \not\equiv 0$ (we also notice that C' is multilinear iff C is). □

[6] It is not difficult to compute $\mathrm{var}(C)$ given Theorem 2 and in fact it is implicit in the proof of the theorem.

3.1 Some Corollaries

An immediate consequence of Theorem 1 is that there are efficient algorithms for polynomial decomposition in circuit classes for which efficient PIT algorithms are known. The proof of the following corollary is immediate given the state of the art PIT algorithms.

Corollary 2. *Let $f(\bar{x})$ be a polynomial. We obtain the following algorithms.*

1. *If f has degree d and m monomials then there is a polynomial time (in m, n, d) black-box algorithm for computing the indecomposable factors of f (this is the circuit class of sparse polynomials, see e.g. [12]).*
2. *If f is computed by a depth-3 circuit with top fan-in k (i.e. a $\Sigma\Pi\Sigma(k)$ circuit, see [17]) and degree d then there is an $(nd)^{\mathcal{O}(k^2)}$ non black-box algorithm for computing the indecomposable factors of f (see [18]). In the black-box model there is an $n^{\mathcal{O}(k^6 \log d)}$ time algorithm over finite fields and an $(nd)^{\mathcal{O}(k^4)}$ time algorithm over $\mathbb{Q}$, for the task (see [28]).*
3. *If f is computed by sum of k Preprocessed Read-Once arithmetic formulas of individual degrees at most d (see [26]), then there is an $(nd)^{\mathcal{O}(k^2)}$ non black-box algorithm for computing the indecomposable factors of f and an $(nd)^{\mathcal{O}(k^2+\log n)}$ black-box algorithm for the problem.*

We now prove Corollary 1. We give a more formal statement, again, in full generality, so that it can be applied to restricted models of arithmetic circuits as well.

Corollary 1 restated: *Let $\mathcal{C}$ be an arithmetic circuit class computing multilinear polynomials. Assume that there is a deterministic PIT algorithm that runs in time $T(s)$ when given as input a circuit of the form $C = C_1 + C_2 \times C_3$, where all the C_i-$s \in \mathcal{C}$ are n-variate circuits of size s and C_2 and C_3 are variable disjoint. Then, there is a deterministic algorithm that when given access (explicit or via a black-box) to an n-variate circuit $C' \in \mathcal{C}$, of size s, runs in time* $\mathrm{poly}(n, T(s))$ *and outputs the* irreducible *factors, $h_1, \ldots, h_k$, of the polynomial computed by C'. Moreover, each h_i can be computed by a size s circuit from $\mathcal{C}$.*

Conversely, assume there is a deterministic factoring algorithm that runs in time $T(s)$ when given as input a size s circuit from $\mathcal{C}$ (or even just a deterministic algorithm for the corresponding decision problem). Then $\mathcal{C}$ has a PIT algorithm, for size s circuits, of running time $\mathcal{O}(T(s+2))$.

In particular, if one of the problems has a polynomial time algorithms, namely $T(s) = \mathrm{poly}(s)$, then so does the other. The two directions hold both in the black-box and non black-box models.

Proof. The claim is immediate from Theorem 1 and Observations 1 and 2. □

4 Concluding Remarks

We showed a strong relation between PIT and polynomial decomposition. As noted, for multilinear polynomials, decomposition is the same as factoring. Thus,

for multilinear polynomials PIT and factorization are equivalent up to a polynomial overhead. It is an interesting question whether such a relation holds for general polynomials. Namely, whether PIT is equivalent to polynomial factorization.

We note that in restricted models it may be the case that a polynomial and one of its factors will have a different complexity. For example, consider the polynomial $f(x_1,\ldots,x_k) = \prod_{i=1}^{k}(x_i^k - 1) + \prod_{i=1}^{k}(x_i - 1) = \prod_{i=1}^{k}(x_i - 1) \cdot \left(\prod_{i=1}^{k}(x_i^{k-1} + \ldots + 1) + 1\right)$. Then f has $2^{k+1} - 1$ monomials, but one of its irreducible factors has k^k monomials. Thus, for $k = \log n$ we can compute f as a sparse polynomial, but some of its factors will not be sparse (the fact that f has only $\log n$ variables is not really important as we can multiply f by $x_{\log n+1} \cdot \ldots \cdot x_n$ and still have the same problem). Thus, it is also interesting to understand whether it is even possible to have some analog of our result for the factorization problem in restricted models. This question touches of course the interesting open problem of whether the depth of a factor can increase significantly with respect to the depth of the original polynomial.

References

1. Kabanets, V., Impagliazzo, R.: Derandomizing polynomial identity tests means proving circuit lower bounds. Computational Complexity 13(1-2), 1–46 (2004)
2. Shpilka, A., Volkovich, I.: Read-once polynomial identity testing. In: Proceedings of the 40th Annual STOC, pp. 507–516 (2008)
3. Gathen, J.v.z., Gerhard, J.: Modern computer algebra. Cambridge University Press, Cambridge (1999)
4. Kaltofen, E.: Polynomial factorization: a success story. In: ISSAC, pp. 3–4 (2003)
5. Gathen, J.v.z.: Who was who in polynomial factorization. In: ISSAC, vol. 2 (2006)
6. Kayal, N.: Derandomizing some number-theoretic and algebraic algorithms. PhD thesis, Indian Institute of Technology, Kanpur, India (2007)
7. Agrawal, M., Kayal, N., Saxena, N.: Primes is in P. Annals of Mathematics 160(2), 781–793 (2004)
8. Mulmuley, K., Vazirani, U., Vazirani, V.: Matching is as easy as matrix inversion. Combinatorica 7(1), 105–113 (1987)
9. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. JACM 27(4), 701–717 (1980)
10. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Symbolic and algebraic computation, pp. 216–226 (1979)
11. DeMillo, R.A., Lipton, R.J.: A probabilistic remark on algebraic program testing. Inf. Process. Lett. 7(4), 193–195 (1978)
12. Klivans, A., Spielman, D.: Randomness efficient identity testing of multivariate polynomials. In: Proceedings of the 33rd Annual STOC, pp. 216–223 (2001)
13. Agrawal, M., Biswas, S.: Primality and identity testing via chinese remaindering. JACM 50(4), 429–443 (2003)
14. Lipton, R.J., Vishnoi, N.K.: Deterministic identity testing for multivariate polynomials. In: Proceedings of the 14th annual SODA, pp. 756–760 (2003)

15. Agrawal, M.: Proving lower bounds via pseudo-random generators. In: Sarukkai, S., Sen, S. (eds.) FSTTCS 2005. LNCS, vol. 3821, pp. 92–105. Springer, Heidelberg (2005)
16. Raz, R., Shpilka, A.: Deterministic polynomial identity testing in non commutative models. Computational Complexity 14(1), 1–19 (2005)
17. Dvir, Z., Shpilka, A.: Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. SIAM J. on Computing 36(5), 1404–1434 (2006)
18. Kayal, N., Saxena, N.: Polynomial identity testing for depth 3 circuits. Computational Complexity 16(2), 115–138 (2007)
19. Arvind, V., Mukhopadhyay, P.: The monomial ideal membership problem and polynomial identity testing. In: Proceedings of the 18th ISAAC, pp. 800–811 (2007)
20. Karnin, Z.S., Shpilka, A.: Deterministic black box polynomial identity testing of depth-3 arithmetic circuits with bounded top fan-in. In: Proceedings of the 23rd Annual CCC, pp. 280–291 (2008)
21. Saxena, N., Seshadhri, C.: An almost optimal rank bound for depth-3 identities. In: Proceedings of the 24th annual CCC (2009)
22. Dvir, Z., Shpilka, A., Yehudayoff, A.: Hardness-randomness tradeoffs for bounded depth arithmetic circuits. SIAM J. on Computing 39(4), 1279–1293 (2009)
23. Agrawal, M., Vinay, V.: Arithmetic circuits: A chasm at depth four. In: Proceedings of the 49th Annual FOCS, pp. 67–75 (2008)
24. Arvind, V., Mukhopadhyay, P.: Derandomizing the isolation lemma and lower bounds for circuit size. In: Goel, A., Jansen, K., Rolim, J.D.P., Rubinfeld, R. (eds.) APPROX and RANDOM 2008. LNCS, vol. 5171, pp. 276–289. Springer, Heidelberg (2008)
25. Kayal, N., Saraf, S.: Blackbox polynomial identity testing for depth 3 circuits. Electronic Colloquium on Computational Complexity (ECCC), (32) (2009)
26. Shpilka, A., Volkovich, I.: Improved polynomial identity testing for read-once formulas. In: APPROX-RANDOM, pp. 700–713 (2009)
27. Karnin, Z.S., Mukhopadhyay, P., Shpilka, A., Volkovich, I.: Deterministic identity testing of depth 4 multilinear circuits with bounded top fan-in. In: Proceedings of the 42th Annual STOC (2010)
28. Saxena, N., Seshadhri, C.: From sylvester-gallai configurations to rank bounds: Improved black-box identity test for depth-3 circuits. Electronic Colloquium on Computational Complexity (ECCC) (013) (2010)
29. Heintz, J., Schnorr, C.P.: Testing polynomials which are easy to compute (extended abstract). In: Proceedings of the 12th annual STOC, pp. 262–272 (1980)
30. Raz, R.: Multi-linear formulas for permanent and determinant are of super-polynomial size. In: Proceedings of the 36th Annual STOC, pp. 633–641 (2004)
31. Raz, R.: Multilinear $NC_1 \neq$ Multilinear NC_2. In: Proceedings of the 45th Annual FOCS, pp. 344–351 (2004)
32. Raz, R., Shpilka, A., Yehudayoff, A.: A lower bound for the size of syntactically multilinear arithmetic circuits. SIAM J. on Computing 38(4), 1624–1647 (2008)
33. Raz, R., Yehudayoff, A.: Lower bounds and separations for constant depth multilinear circuits. In: IEEE Conference on Computational Complexity, pp. 128–139 (2008)
34. Hancock, T.R., Hellerstein., L.: Learning read-once formulas over fields and extended bases. In: Proceedings of the 4th Annual COLT, pp. 326–336 (1991)
35. Bshouty, N.H., Hancock, T.R., Hellerstein, L.: Learning arithmetic read-once formulas. SIAM J. on Computing 24(4), 706–735 (1995)
36. Kaltofen, E., Koiran, P.: On the complexity of factoring bivariate supersparse (lacunary) polynomials. In: ISSAC, pp. 208–215 (2005)

On Sums of Roots of Unity

Bruce Litow*

Discipline of IT, James Cook University, Townsville, Qld. 4811, Australia

Abstract. We make two remarks on linear forms over $\mathcal{Z}$ in complex roots of unity. First we show that a Liouville type lower bound on the absolute value of a nonvanishing form can be derived from the time complexity upper bound on Tarski algbera. Second we exhibit an efficient randomized algorithm for deciding whether a given form vanishes. In the special case where the periods of the roots of unity are mutually coprime, we can eliminate randomization. This efficiency is surprising given the doubly exponential smallness of the Liouville bound.

1 Introduction

We make two remarks on linear forms over $\mathcal{Z}$ in complex roots of unity. First we show that a Liouville type lower bound on the absolute value of a nonvanishing form can be derived from the time complexity upper bound on Tarski algbera. Second we exhibit an efficient randomized algorithm for deciding whether a given form vanishes. In the special case where the periods of the roots of unity are mutually coprime, we can eliminate randomization. This efficiency is surprising given the doubly exponential smallness of the Liouville bound.

In general, a linear form over $\mathcal{Z}$ is an expression $\Lambda(X_1, \ldots, X_r) = \sum_{i=1}^{r} b_i \cdot X_i$, where $b_i \in \mathcal{Z}$ and X_i is an indeterminate. We are interested in the case where the X_i range over complex roots of unity. The period d of a root of unity ω is the least positive integer such that $\omega^d = 1$. In Section 2 we give an efficient algorithm to decide whether $\Lambda(\omega_1, \ldots, \omega_r) = 0$. We refer to this as the sums problem. We explain what we mean by efficient. Let $b = \sum_{i=1}^{r} |b_i|$ and $\ell = \log b$ (base 2 logarithm). Let d_i be the period of ω_i and $d = \mathrm{LCM}(d_1, \ldots, d_r)$. We regard $\ell + \log d$ as the size of an instance. As usual, $\phi(x)$ denotes the Euler totient of x. We show that the sums problem is in **RP** (random polytime) and if the d_i are mutually coprime, then we show that the sums problem is in **P** (polytime). Details about **RP** and **P** can be found in [1]. Determining whether these forms vanish is arguably the most elementary problem of computational algebraic number theory but we do not know of other methods that have polynomial running time in both the bit sizes of b and d.

We introduce some notation. The few facts from algebra that we use later can be found in [4,5]. $\mathcal{Z}, \mathcal{N}$ and $\mathcal{N}_+$ denote the integers, nonnegative integers and positive integers, respectively. $\mathcal{Q}$ and $\mathcal{C}$ denote the complex numbers and

* Discipline of IT, James Cook University, Townsville, Qld. 4811, Australia
bruce.litow@jcu.edu.au

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 420–425, 2010.

rationals, respectively. $\mathcal{Z}[X]$ denotes the polynomials over $\mathcal{Z}$ in the indeterminate X. If $f \in \mathcal{Z}[X]$ and $\alpha \in \mathcal{C}$, then $f(\alpha)$ is the value of $f(X)$ at $X = \alpha$. The coefficient of X^n in $f \in \mathcal{Z}[X]$ is denoted by $[n]f(X)$. For $f(X) \in \mathcal{Z}[X]$ let $|f(X)| = \sum_n |[n]f(X)|$. We regard $\log |f(X)|$ as the size of $f(X)$.

2 Tarski Algebra and Liouville Bounds

A detailed study of Liouville bounds and related subjects can be found in [8]. The Tarski algebra time complexity result is taken from [2]. If $f(X) \in \mathcal{Z}[X]$, $\alpha \in \mathcal{C}$ is algebraic and $f(\alpha) \neq 0$, then one can determine a lower bound on $|f(\alpha)|$ that depends only on $|f(X)|$ and algebraic properties of α. This is known as a Liouville bound. We next relate the sums problem to Liouville bounds.

Consider a linear form $\Lambda(\omega_1, \ldots, \omega_r)$. Let ω be a root of unity of period d. We can write $\omega_i = \omega^{c_i}$ where $c_i \in \mathcal{N}_+$ and $c_i \leq d$. From this we see that $\Lambda(\omega_1, \ldots, \omega_r) = \sum_{i=1}^{r} b_i \cdot \omega^{c_i}$. Let $g(X) = \sum_{i=1}^{r} b_i \cdot X^{c_i}$. That is, $\Lambda(\omega_1, \ldots, \omega_r) = g(\omega)$. This enables us to apply directly a Liouville bound on $g(\omega)$ to $\Lambda(\omega_1, \ldots, \omega_r)$. We fix the polynomial $g(X)$ in this role. Note that $b = |g(X)|$.

We employ the first order theory of the real field, $\mathcal{R}$, known as Tarski algebra (TA) to simplify proofs and also to reveal certain concepts that would otherwise be obscured by approximation methods tailored to quite specific problems concerning algebraic numbers. The original results, due to A. Tarski were reported in a preliminary way in [6] and then more comprehensively in [7]. Tarski showed that any sentence of TA could be decided by explicit quantifier elimination. Subsequent to his demonstration others refined the decision method, thereby vastly reducing the time complexity over Tarski's procedure. The decision problem for TA is treated in great detail in [2].

For our purpose a TA formula has the form

$$Q_1 y_1, \ldots, Q_a y_a \; A(x_1, \ldots, x_b, y_1, \ldots, y_a) \; ,$$

where the variables $y_1, \ldots, y_a$ are bound by quantifiers $Q_1, \ldots, Q_a$ (existential or universal), $x_1, \ldots, x_b$ are free variables and $A(x_1, \ldots, x_b, y_1, \ldots, y_a)$ is a Boolean expression in atomic formulas. An atomic formula has the form $u(x_1, \ldots, x_b, y_1, \ldots, y_a) \diamond 0$, where

$$u(x_1, \ldots, x_b, y_1, \ldots, y_a) \in \mathcal{Z}[x_1, \ldots, x_b, y_1, \ldots, y_a]$$

and $\diamond$ is one of the relation symbols $=, <, >$. This application of triality eliminates the need for negation in TA formulas. The coefficients of $u(x_1, \ldots, x_b, y_1, \ldots, y_a)$ are understood to be built up from the 0-adic function symbols $0, 1$ in binary notation (essentially). The axioms of TA are the field axioms for $\mathcal{R}$ and the standard first order logic axioms with equality. Since we need to work over $\mathcal{C}$ we simply note, as Tarski did in his original work that arithmetic over $\mathcal{C}$ can be faithfully represented by arithmetic on elements of $\mathcal{R} \times \mathcal{R}$ supplemented by the Fortran-style rules for complex arithmetic in terms of ordered pairs of reals. Such things as $|\alpha|^2$ for $\alpha \in \mathcal{C}$ admit straightforward representations in this way.

We write $B(x_1, \ldots, x_b)$ to indicate a TA formula, possibly with bound variables and the indicated free variables. We say that $B(x_1, \ldots, x_b)$ and $C(x_1, \ldots, x_b)$ are equivalent if for all $(\alpha_1, \ldots, \alpha_b) \in \mathcal{R}^b$, $B(\alpha_1, \ldots, \alpha_b) \Leftrightarrow C(\alpha_1, \ldots, \alpha_b)$. The size of a formula is defined naturally in terms of the sizes of the polynomials in atomic fromulas.

The next theorem is established in [2].

Theorem 1. *Given $B(x_1, \ldots, x_b)$, an equivalent, quantifier-free formula can be computed in $a^{d^{O(c)}}$ time, where a is the size of $B(x_1, \ldots, x_b)$, d is the number of distinct variables (including bound ones) in $B(x_1, \ldots, x_b)$ and c is the number of quantifier alternations.*

Using Theorem 1 we now derive a lower bound on $|\Lambda(\omega_1, \ldots, \omega_r)|$ in case it does not vanish . Recall that $\ell = \log b = \log |g(X)|$.

Theorem 2. *If $\Lambda(\omega_1, \ldots, \omega_r) \neq 0$, then*

$$|\Lambda(\omega_1, \ldots, \omega_r)| > 1/2^{(d+\ell)^a} ,$$

where a is an absolute positive constant.

Proof. For the moment we proceed informally outside TA. With variables ranging over $\mathcal{C}$. We introduce formulas in these variables.

- $A(x) \Leftrightarrow x^d = 1 \wedge g(x) \neq 0$.
- $B(x, y) \Leftrightarrow A(x) \wedge |g(x)|^2 \geq y$. We could represent $|g(x)|$ in TA but we are aiming for a simple exposition.
- $C(z) \Leftrightarrow \forall\, x, y\ B(x, y) \wedge B(x, z) \Rightarrow y \leq z$.

It is plain that $C(\alpha)$ holds for exactly one value, namely the least value of $|g(\omega)|$ over all d-th roots of unity for which $g(\omega) \neq 0$.

Conversion of $C(z)$ to a TA formula is straightforward. For example x^d is actually the binomial expansion of $(u + \sqrt{-1} \cdot v)^d$, with real and imaginary terms collected separately. A similar observation applies to rendering $g(x)$. Note that we can use $g(x) \cdot g(\bar{x}) = |g(x)|^2$, where the bar indicates complex conjugation. It is easy to check that a TA formula for $C(z)$ of size $(d+\ell)^{O(1)}$ can be constructed. For example, $g(X)$ requires $O(\ell \cdot d)$ bits. Note that the binomial coefficients can be represented in TA in binary notation. Applying Theorem 1, $C(z)$ is equivalent to a quantifier-free TA formula $\tilde{C}(z)$, which can be constructed in $(d + \ell)^a$ time for some absolute positive constant a. Now, $\tilde{C}(z)$ is a Boolean in atomic formulas. We can write $\tilde{C}(z)$ in DNF as $\bigvee_p \bigwedge_q D_{p,q}$ where each $D_{p,q}$ is atomic. The time required to do this is irrelevant. The critical size element has to do with the $D_{p,q}$. From the toplogy of $\mathcal{C}$ and the fact that $\tilde{C}(z)$ holds for exactly one value α we conclude that α is a zero of some of the polynomials appearing among the $D_{p,q}$. Each polynomal, since it was constructed in $(d + \ell)^a$ time has bit size at most $(d + \ell)^a$. This means that the coefficient sizes of the polynomial are bounded above by $2^{(d+\ell)^a}$. It is elementary from algebra that the zero $|\alpha|$ is bounded below by $1/2^{2 \cdot (d+\ell)^{2a}}$.

We compare the TA derived lower bound with a standard Liouville bound. For details on Liouville bound results see [8].

Theorem 3. *If $g(X) \in \mathcal{Z}[X]$ and α is a root of unity of period d such that $g(\alpha) \neq 0$, then $|g(\alpha)| > |g(X)|^{-d}$.*

Our TA derived lower bound is $2^{-(d+\ell)^a}$ versus the Liouville bound of $b^{-d} = 2^{-d \cdot \ell}$. The constant $a > 2$, so the TA bound is inferior to the Liouville bound. However, the TA bound is much easier to derive. We do not know of a sharp lower bound on $|\Lambda(\omega_1, \ldots, \omega_r)|$.

3 Algorithms for the Sums Problem

We retain all notation from Section 1. Recall that $\ell + \log b$ is the sums problem instance size. Let $\mathbf{e}(z) = \exp(2 \cdot \pi \cdot \sqrt{-1} \cdot z)$. Let $[a : b]$ denote the integer interval between a and b inclusive. For $x, y, z \in \mathcal{N}$ let $|x|_y$ be the least $z \equiv x \bmod y$.

The next lemma enables us to sidestep the Liouville lower bound.

Lemma 1. *Let $g(X) \in \mathcal{Z}[X]$. If ω is a root of unity of period d such that $g(\omega) \neq 0$, then for at least $\phi(d)/2$ elements $k \in [1 : d-1]$, $|g(\omega^k)| > 1/|g(X)|$.*

Proof. If k is coprime to d, then $g(\omega) = 0 \Leftrightarrow g(\omega^k) = 0$. This follows from the fact that ω and ω^k have the same minimal polynomial $u(X)$ so if $g(\omega) = 0$, then $g(X)$ is divisible by $\beta \cdot u(X)$, where β is a nonzero rational. Assume $g(\omega) \neq 0$. Let $A = \prod_k g(\omega^k)$, where $k \in [1 : d-1]$ runs over all elements that are coprime to d. We argue that A is a nonnegative integer. Since $A \neq 0$, it remains to show that A is an integer.

- The roots of unity form a subset of the ring of algebraic integers. It follows that A is an algebraic integer.
- A is fixed by all automorphisms of $\mathcal{Q}[\omega]$. These are given by $\omega \rightarrow \omega^k$ for k coprime to d. Hence, $A \in \mathcal{Q}$.
- We have that A is in the intersection of the algebraic integers and $\mathcal{Q}$ but this is $\mathcal{Z}$.

We now know that $|A| \geq 1$. Now, $|g(\omega^k)| \leq |g(X)|$. Assume that c factors of A have absolute value less than $1/|g(X)|$. We must have, at least, $|g(X)|^{\phi(d)-c}/|g(X)|^c \geq 1$. This forces $c < \phi(d)/2$.

Since $\Lambda(\omega_1, \ldots, \ldots, \omega_r) = g(\omega)$ we also have $\Lambda(\omega_1^k, \ldots, \omega_r^k) = g(\omega^k)$. This tells us that if k and d are coprime, then $\Lambda(\omega_1, \ldots, \ldots, \omega_r) = 0 \Leftrightarrow \Lambda(\omega_1^k, \ldots, \omega_r^k) = 0$.

Theorem 4. *The sums problem is in* **RP**.

Proof. Uniformly and randomly choose $k \in [1 : d-1]$. The probability that k is coprime to d is $\phi(d)/(d-1)$ so the probability that k also satisfies $|\Lambda(\omega)1^k, \ldots, \omega_r^k)| > 1/b$ is by Lemma 1 at least $\phi(d)/(2d) = \Omega(1/\ell)$. This follows from the prime number theorem since $\phi(d)$ is at least the number of primes up to d, which is

$\Theta(d/\log d)$. See [3]. Note that $\Lambda(\omega_1^k,\ldots,\omega_r^k) = \Lambda(\omega_1^{|k|_{d_1}},\ldots,\omega_r^{|k|_{d_r}})$. Let $E_m(z) = \sum_{n=0}^{m}(2\cdot\pi\cdot\sqrt{-1}\cdot z)^n/n!$. Obtain $\hat{\Lambda}$ from $\Lambda(\omega_1^{|k|_{d_1}},\ldots,\omega_r^{|k|_{d_r}})$ by substituting $E_m(|g_i\cdot k|_{d_i}/d_i)$ for ω_i^k, where $\omega_i = \mathbf{e}(g_i/d_i)$. It is easy to check that if $m > 4\cdot\pi\cdot\exp(1)$, then $|\omega_i^k - E_m(|g_i\cdot k|_{d_i}/d_i)| < 1/2^m$. If $m > \max\{1+\log b, 4\cdot\pi\cdot\exp(1)\}$, then

$$|\Lambda(\omega_1^k,\ldots,\Lambda_r^k) - \hat{\Lambda}| < 1/(2b)\ . \tag{1}$$

If $|\hat{\Lambda}| \geq 1/(2b)$, then $\Lambda(\omega_1,\ldots,\omega_r) \neq 0$ because otherwise $|\hat{\Lambda} - \Lambda(\omega_1^k,\ldots,\Lambda_r^k)| = |\hat{\Lambda}| < 1/(2b)$, contradicting Eq. 1. If $|\hat{\Lambda}| < 1/(2b)$, then we claim the probability that $\Lambda(\omega_1,\ldots,\omega_r) \neq 0$ is less than $1 - \Theta(1/\ell)$. This follows from the fact that if $|\hat{\Lambda}| > 1/b$, then $|\hat{\Lambda} - \Lambda(\omega_1^k,\ldots,\Lambda_r^k)| > 1/(2b)$, contradicting Eq. 1. The probability that $|\hat{\Lambda}| > 1/b$ is $\Omega(1/\ell)$. These probabilities satisfy the definition of **RP** since $0 \leq (1-1/\ell)^\ell < 1/\exp(1)$.

We modify notation from the proof of Theorem 4. Now, define $\Lambda = \sum_k \Lambda(\omega_1^k,\ldots,\omega_r^k)\cdot\Lambda(\omega_1^{d-k},\ldots,\omega_r^{d-k})$, where $k \in [1:d-1]$ is coprime to d. Since $g(\omega^k)$ is the complex conjugate of $g(\omega^{d-k})$, it is clear that $\Lambda = \sum_k |\Lambda(\omega_1^k,\ldots,\omega_r^k)|^2$.

Lemma 2. *If $\Lambda(\omega_1,\ldots,\omega_r) = 0$, then $\Lambda = 0$, otherwise $\Lambda > \phi(d)/(2b^2)$.*

Proof. The first claim follows from $\Lambda(\omega_1^k,\ldots,\omega_r^k) = g(\omega^k)$. The second claim follows from Lemma 1.

Theorem 5. *If the d_i are mutually coprime, then the sums problem is in* **P**.

Proof. Assume the d_i are mutually coprime. Let U be the set of $k \in [1:d-1]$ that are coprime to d. By the Chinese remainder theorem, the elements of U are in bjiective correspondence with the set of tuples $(a_1,\ldots,a_r)$ where $a_i \in [1:d_i-1]$. Assume $i \neq j$. Let $\mu_{i,j} = \prod_{h\neq i,j}(d_h-1)$ and $\mu_{i,i} = \prod_{j\neq i}(d_j-1)$. For each ordered pair $(a,b) \in [1:d_i-1]\times[1:d_j-1]$ there are $\mu_{i,j}$ elements $k \in U$ such that $(|k|_{d_i}, |d-k|_{d_j}) = (a,b)$. If $i = j$, then for each $(a, d-a) \in [1:d_i-1]\times[1:d_i-1]$ there are $\mu_{i,i}$ elements $k \in U$ such that $(|k|_{d_i}, |d-k|_{d_i}) = (a, d-a)$.

It follows from the foregoing observations that

$$\Lambda = \sum_{i\neq j} b_i\cdot b_j\cdot\mu_{i,j}\cdot \sum_{a\in[1:d_i-1],b\in[1:d_j-1]} \omega_i^a\cdot\omega_j^b + \sum_i b_i^2\cdot\mu_{i,i}\cdot(d_i-1);\,.$$

We have used $\omega_i^a\cdot\omega_i^{d-a} = 1$ and $\sum_{a\in[1:d_i-1]} 1 = d_i - 1$.

Noting that

$$\omega_i^a\cdot\omega_j^b = \mathbf{e}(\frac{|a\cdot d_j + b\cdot d_i|_{d_i\cdot d_j}}{d_i\cdot d_j})$$

we introduce $\hat{\Lambda}$ by substituting $E_m(\frac{|a\cdot d_j+b\cdot d_i|_{d_i\cdot d_j}}{d_i\cdot d_j})$ for $\mathbf{e}(\frac{|a\cdot d_j+b\cdot d_i|_{d_i\cdot d_j}}{d_i\cdot d_j})$ in Λ.

Let $d_* = \max d_i$. There are fewer than $d_*^2 + d_* < 2\cdot d^2$ terms in our sum. The maximum value of $|b_i\cdot b_j\cdot\mu_{i,j}$ is less than $b^2\cdot d$. From this we have

$$|\Lambda - \hat{\Lambda}| < 2\cdot(b\cdot d)^2\cdot\delta\ ,$$

where δ is the maximum over all

$$|\mathbf{e}(\frac{|a \cdot d_j + b \cdot d_i|_{d_i \cdot d_j}}{d_i \cdot d_j}) - E_m(\frac{|a \cdot d_j + b \cdot d_i|_{d_i \cdot d_j}}{d_i \cdot d_j})| \ .$$

From the proof of Theorem 4 if $m > \max\{1 + \log b, 4 \cdot \pi \cdot \exp(1)\}$, then $\delta < 1/2^m$. By choosing m so that $2 \cdot (b \cdot d)^2/2^m < 1/(2b^2)$ we can determine whether $\Lambda = 0$ since if $\Lambda \neq 0$ we know that $|\Lambda| > \phi(d)/(2b^2) > 1/b^2$. Thus, we can choose $m > \max\{4 \cdot \pi \cdot \exp(1), \log(4 \cdot b^3 \cdot d^2)\}$. From this bound it is clear that $\hat{\Lambda}$ can be computed in $(\ell + \log d)^{O(1)}$ time.

We made several overestimates. Slightly better bounds may result in cases where a quantitative relationship between d and b holds.

References

1. Balcazar, J.L., Diaz, J., Gabarro, J.: Structural Complexity. Springer, Heidelberg (1988)
2. Basu, S., Pollack, R., Roy, M.-F.: Algorithms in Real Algebraic Geometry. Springer, Heidelberg (2003)
3. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers. Oxford Press, USA (1979)
4. Lang, S.: Algebra. Addison-Wesley, Reading (1965)
5. Samuel, P.: Algebraic Theory of Numbers. Hermann (1970)
6. Tarski, A.: Sur les ensembles définissables de nombres réels. Fundamenta Mathematicae 17, 210–239 (1931)
7. Tarski, A.: A decision method for elementary algebra and geometry. Technical report, Rand Corp. (1948)
8. Waldschmidt, M.: Linear independence of logarithms of algebraic numbers. Technical Report IMSc 116, Inst. of Math. Science, Madras (1992)

Exponential Time Complexity of the Permanent and the Tutte Polynomial
(Extended Abstract)

Holger Dell[1,⋆], Thore Husfeldt[2], and Martin Wahlén[3]

[1] Humboldt University of Berlin, Germany
[2] IT University of Copenhagen, Denmark and Lund University, Sweden
[3] Lund University, Sweden and Uppsala University, Sweden

Abstract. The Exponential Time Hypothesis (ETH) says that deciding the satisfiability of n-variable 3-CNF formulas requires time $\exp(\Omega(n))$. We relax this hypothesis by introducing its counting version #ETH, namely that every algorithm that counts the satisfying assignments requires time $\exp(\Omega(n))$. We transfer the sparsification lemma for d-CNF formulas to the counting setting, which makes #ETH robust.

Under this hypothesis, we show lower bounds for well-studied #P-hard problems: Computing the permanent of an $n \times n$ matrix with m nonzero entries requires time $\exp(\Omega(m))$. Restricted to 01-matrices, the bound is $\exp(\Omega(m/\log m))$. Computing the Tutte polynomial of a multigraph with n vertices and m edges requires time $\exp(\Omega(n))$ at points (x, y) with $(x-1)(y-1) \neq 1$ and $y \notin \{0, \pm 1\}$. At points $(x, 0)$ with $x \notin \{0, \pm 1\}$ it requires time $\exp(\Omega(n))$, and if $x = -2, -3, \ldots$, it requires time $\exp(\Omega(m))$. For simple graphs, the bound is $\exp(\Omega(m/\log^3 m))$.

1 Introduction

The permanent of a matrix and the Tutte polynomial of a graph are central topics in the study of counting algorithms. Originally defined in the combinatorics literature, they unify and abstract many enumeration problems, including immediate questions about graphs such as computing the number of perfect matchings, spanning trees, forests, colourings, certain flows and orientations, but also less obvious connections to other fields, such as link polynomials from knot theory, reliability polynomials from network theory, and (maybe most importantly) the Ising and Potts models from statistical physics.

From its definition (repeated in (1) below), the permanent of an $n \times n$-matrix can be computed in $O(n!n)$ time, and the Tutte polynomial (2) can be evaluated in time exponential in the number of edges. Both problems are famously #P-hard, which rules out the existence of polynomial-time algorithms under standard complexity-theoretic assumptions, but that does not mean that we have to resign ourselves to brute-force evaluation of the definition. In fact, Ryser's

⋆ Supported by the Deutsche Forschungsgemeinschaft within the research training group "Methods for Discrete Structures" (GRK 1408).

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 426–437, 2010.

famous formula [19] computes the permanent with only $\exp(O(n))$ arithmetic operations, and more recently, an algorithm with running time $\exp(O(n))$ for n-vertex graphs has also been found [4] for the Tutte polynomial. Curiously, both of these algorithms are based on the inclusion–exclusion principle. We show that these algorithms cannot be significantly improved, by providing conditional lower bounds of $\exp(\Omega(n))$ for both problems.

It is clear that #P-hardness is not the right conceptual framework for such claims, as it is unable to distinguish between different types of super-polynomial time complexities. For example, the Tutte polynomial for planar graphs remains #P-hard, but can be computed in time $\exp(O(\sqrt{n}))$ [20]. Therefore, we work under the *Exponential Time Hypothesis* (ETH), viz. the complexity theoretic assumption that *some* hard problem (namely, Satisfiability of 3-CNF formulas in n variables) requires time $\exp(\Omega(n))$. More specifically, we introduce #ETH, a counting analogue of ETH which models the hypothesis that *counting* the satisfying assignments requires time $\exp(\Omega(n))$.

Computing the permanent. The permanent of an $n \times n$ matrix A is defined as

$$\operatorname{per} A = \sum_{\pi \in S_n} \prod_{1 \le i \le n} A_{i\pi(i)}\,, \tag{1}$$

where S_n is the set of permutations of $\{1, \ldots, n\}$. This is redolent of the determinant from linear algebra, $\det A = \sum_\pi \operatorname{sign}(\pi) \prod_i A_{i\pi(i)}$, the only difference is an easily computable sign for every summand. Both definitions involve a summation with $n!$ terms, but admit much faster algorithms that are textbook material: The determinant can be computed in polynomial time using Gaussian elimination and the permanent can be computed in $O(2^n n)$ operations using Ryser's formula.

Valiant's celebrated #P-hardness result [23] for the permanent shows that no polynomial-time algorithm à la "Gaussian elimination for the permanent" can exist unless P = NP, and indeed unless $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$. Several unconditional lower bounds for the permanent in restricted models of computation are also known. Jerrum and Snir [13] have shown that monotone arithmetic circuits need $n(2^{n-1} - 1)$ multiplications to compute the permanent, a bound they can match with a variant of Laplace's determinant expansion. Raz [18] has shown that multi-linear arithmetic formulas for the permanent require size $\exp(\Omega(\log^2 n))$. Ryser's formula belongs to this class of formulas, but is much larger than the lower bound; no smaller construction is known. Intriguingly, the same lower bound holds for the determinant, where it is matched by a formula of size $\exp(O(\log^2 n))$ due to Berkowitz [2]. One of the easy consequences of the present results is that Ryser's formula is in some sense optimal under #ETH. In particular, no uniformly constructible, subexponential size formula such as Berkowitz's can exist for the permanent unless #ETH fails.

A related topic is the expression of $\operatorname{per} A$ in terms of $\det f(A)$, where $f(A)$ is a matrix of constants and entries from A and is typically much larger than A. This question has fascinated many mathematicians for a long time, see Agrawal's survey [1]; the best known bound on the dimension of $f(A)$ is $\exp(O(n))$ and it

is conjectured that all such constructions require exponential size. In particular, it is an important open problem if a permanent of size n can be expressed as a determinant of size $\exp(O(\log^2 n))$. Our result is that under #ETH, if such a matrix $f(A)$ exists, computing f must take time $\exp(\Omega(n))$.

Computing the Tutte polynomial. The Tutte polynomial, a bivariate polynomial associated with a given graph $G = (V, E)$ with n vertices and m edges, is defined as

$$T(G; x, y) = \sum_{A \subseteq E} (x-1)^{k(A)-k(E)} (y-1)^{k(A)+|A|-|V|}, \tag{2}$$

where $k(A)$ denotes the number of connected components of the subgraph (V, A).

Despite their unified definition (2), the various computational problems given by $T(G; x, y)$ for different points (x, y) differ widely in computational complexity, as well as in the methods used to find algorithms and lower bounds. For example, $T(G; 1, 1)$ equals the number of spanning trees in G, which happens to admit a polynomial time algorithm, curiously again based on Gaussian elimination. On the other hand, the best known algorithm for computing $T(G; 2, 1)$, the number of forests, runs in $\exp(O(n))$ time.

Computation of the Tutte polynomial has fascinated researchers in computer science and other fields for many decades. For example, the algorithms of Onsager and Fischer from the 1940s and 1960s for computing the so-called partition function for the planar Ising model are viewed as major successes of statistical physics and theoretical chemistry; this corresponds to computing $T(G; x, y)$ along the hyperbola $(x-1)(y-1) = 2$ for planar G. Many serious attempts were made to extend these results to other hyperbolas or graph classes, but "after a quarter of a century and absolutely no progress", Feynman in 1972 observed that "the exact solution for three dimensions has not yet been found".[1]

As for the permanent, the failure of theoretical physics to "solve the Potts model" and sundry other questions implicit in the computational complexity of the Tutte polynomial were explained only with Valiant's #P-hardness programme. After a number of papers, culminating in [12], the polynomial-time complexity of exactly computing the Tutte polynomial at points (x, y) is now completely understood: it is #P-hard everywhere except at those points (x, y) where a polynomial-time algorithm is known; these points consist of the hyperbola $(x-1)(y-1) = 1$ as well as the four points $(1, 1), (-1, -1), (0, -1), (-1, 0)$.

We give an $\exp(\Omega(n))$ lower bound that matches the $\exp(O(n))$ algorithm from [4] and which holds under #ETH everywhere except for $|y| = 1$. In particular, this establishes a gap to the planar case, which admits an $\exp(O(\sqrt{n}))$ algorithm [20]. Our hardness results apply (though not everywhere, and sometimes with a weaker bound) even if the graphs are sparse and simple. These classes are of particular interest because most of the graphs arising from applications in statistical mechanics arise from bond structures, which are sparse and simple.

[1] The Feynman quote and many other quotes describing the frustration and puzzlement of physicists around that time can be found in the copious footnotes of [11].

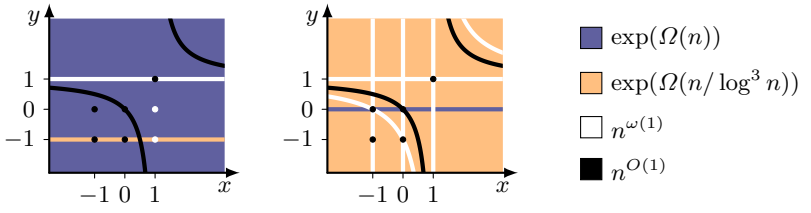


Fig. 1. Exponential time complexity under #ETH of the Tutte plane for multigraphs (left) and simple graphs (right) in terms of n, the number of vertices. White areas on the map correspond to uncharted territory. The black hyperbola $(x-1)(y-1)=1$ and the four points close to the origin are in P. Everywhere else, in the shaded regions, we prove a lower bound exponential in n, or within a polylogarithmic factor of it.

It has been known since the 1970s [16] that graph 3-colouring can be solved in time $\exp(O(n))$, and this is matched by an $\exp(\Omega(n))$ lower bound under ETH [10]. Since graph 3-colouring corresponds to evaluating T at $(-2,0)$, the exponential time complexity for $T(G;-2,0)$ was thereby already understood. In particular, computing $T(G;x,y)$ for input G *and* (x,y) requires vertex-exponential time, an observation that is already made in [7] without explicit reference to ETH.

The literature for computing the Tutte polynomial is very rich, and we make no attempt to survey it here. A recent paper of Goldberg and Jerrum [9], which shows that the Tutte polynomial is even hard to approximate for large parts of the Tutte plane, contains an overview. A list of graph classes for which subexponential time algorithms are known can be found in [4].

2 Results

The exponential time hypothesis (ETH) as defined in [10] is that satisfiability of 3-CNF formulas cannot be computed substantially faster than by trying all possible assignments, i.e., it requires time $\exp(\Omega(n))$. We define the counting exponential time hypothesis via the counting version of 3-Sat.

Name. #3-Sat
Input. 3-CNF formula φ with n variables and m clauses.
Output. The number of satisfying assignments to φ.

The best known algorithm for this problem runs in time $O(1.6423^n)$ [15]. Our hardness results are based on the following hypothesis.

(#ETH) There is a constant $c>0$ such that no deterministic algorithm can compute #3-Sat in time $\exp(c\cdot n)$.

At the expense of making the hypothesis more unlikely, the term "deterministic" may be replaced by "randomized", but we ignore such issues here. Note that ETH trivially implies #ETH whereas the other direction is not known. By introducing

the sparsification lemma, [10] show that ETH is a robust notion in the sense that the clause width 3 and the parameter n in its definition can be replaced by $d \geq 3$ and m, respectively, to get an equivalent hypothesis, albeit the constant c may change in doing so. We transfer the sparsification lemma to #d-Sat and get a similar kind of robustness for #ETH:

Theorem 1. *For all $d \geq 3$, #ETH holds if and only if #d-Sat requires time $\exp(\Omega(m))$.*

In the full paper, we go into more depth about computational complexity aspects of #ETH, including a proof of Thm. 1.

The Permanent. For a set S of rationals we define the following problems:

Name. $\textsc{Perm}^S$
Input. Square matrix A with entries from S.
Output. The value of $\operatorname{per} A$.

We write Perm for $\textsc{Perm}^{\mathbb{N}}$. If B is a bipartite graph with a_{ij} edges from the ith vertex in the left half to the jth vertex in the right half ($1 \leq i, j \leq n$), then $\operatorname{per}(a_{ij})$ equals the number of perfect matchings of B. Thus $\textsc{Perm}^{0,1}$ and Perm and can be viewed as counting the perfect matchings in bipartite graphs and multigraphs, respectively.

We express our lower bounds in terms of m, the number of non-zero entries of A. Without loss of generality, $n \leq m$, so the same bounds hold for the parameter n as well. Note that these bounds imply that the hardest instances have roughly linear density.

Theorem 2. *Under #ETH,*

(i) $\textsc{Perm}^{-1,0,1}$ *requires time* $\exp(\Omega(m))$.
(ii) Perm *requires time* $\exp(\Omega(m))$.
(iii) $\textsc{Perm}^{01}$ *requires time* $\exp(\Omega(m/\log n))$.

The proof is in §3. For (i), we follow a standard reduction by Valiant [23,17] but use a simple equality gadget derived from [5] instead of Valiant's XOR-gadget. For (ii) we use interpolation to get rid of the negative weights. Finally, to establish (iii) we replace large positive weights by gadgets of logarithmic size, which increases the number of vertices and edges by a logarithmic factor.

The Tutte Polynomial. The computational problem $\textsc{Tutte}(x, y)$ is defined for each pair (x, y) of rationals.

Name. $\textsc{Tutte}(x, y)$.
Input. Undirected multigraph G with n vertices.
Output. The value of $T(G; x, y)$.

In general, parallel edges and loops are allowed; we write $\textsc{Tutte}^{01}(x, y)$ for the special case where the input graph is simple.

Our main result is that under #ETH, $\textsc{Tutte}(x, y)$ requires time $\exp(\Omega(n))$ for specific points (x, y), but the size of the bound, and the graph classes for which it holds, varies. We summarise our results in the theorem below, see also Fig. 1. Our strongest reductions give edge-exponential lower bounds, i.e., bounds in terms of the parameter m, which implies the same bound in terms of n because $m \geq n$ in connected graphs. Moreover, a lower bound of $\exp(\Omega(m))$ together with the algorithm in time $\exp(O(n))$ from [4] implies that worst-case instances are *sparse*, in the sense that $m = O(n)$. At other points we have to settle for a vertex-exponential lower bound $\exp(\Omega(n))$. While this matches the best upper bound, it does not rule out a vertex-subexponential algorithm for sparse graphs.

Theorem 3. *Let* $(x, y) \in \mathbb{Q}^2$. *Under #ETH,*

(i) $\textsc{Tutte}(x, y)$ *requires time* $\exp(\Omega(n))$ *if* $(x-1)(y-1) \neq 1$ *and* $y \notin \{0, \pm 1\}$.
(ii) $\textsc{Tutte}^{01}(x, y)$ *requires time* $\exp(\Omega(m/\log^3 m))$ *if* $(x-1)(y-1) \notin \{0, 1, 2\}$ *and* $x \notin \{-1, 0\}$.
(iii) $\textsc{Tutte}^{01}(x, 0)$ *requires time* $\exp(\Omega(m))$ *if* $x \in \{-2, -3, \ldots\}$.
(iv) $\textsc{Tutte}^{01}(x, 0)$ *requires time* $\exp(\Omega(n))$ *if* $x \notin \{0, \pm 1\}$.

In an attempt to prove these results, we may first turn to the literature, which contains a cornucopia of constructions for proving hardness of the Tutte polynomial in various models. In these arguments, a central role is played by graph transformations called thickenings and stretches. A k-thickening replaces every edge by a bundle of k edges, and a k-stretch replaces every edge by a path of k edges. This is used do 'move' an evaluation from one point to another. For example, if H is the 2-stretch of G then $T(H; 2, 2) \sim T(G; 4, \frac{4}{3})$. Thus, every algorithm for $(2, 2)$ works also at $(4, \frac{4}{3})$, connecting the hardness of the two points. These reductions are very well-developed in the literature, and are used in models that are immune to polynomial-size changes in the input parameters, such as #P-hardness and approximation complexity. However, in order to establish our exponential hardness results, we cannot always afford such constructions, otherwise our bounds would be of the form $\exp(\Omega(n^{1/r}))$ for some constant r depending on the blowup in the proof. In particular, the parameter n is destroyed already by a 2-stretch in a nonsparse graph.

The proofs are omitted, though we sketch the construction involved in the proof of Thm. 3 (ii), which may be of independent interest. Where we can, we sample from established methods, carefully avoiding or modifying those that are not parameter-preserving. At other times we require completely new ideas; the constructions in §5, which use Theta graph products instead of thickenings and stretches, may be of independent interest. Like many recent papers, we use Sokal's multivariate version of the Tutte polynomial, which vastly simplifies many of the technical details.

Consequences. The permanent and Tutte polynomial are equivalent to, or generalisations of, various other graph problems, so our lower bounds hold for these problems as well. In particular, it takes time $\exp(\Omega(m))$ to compute the following graph polynomials (for example, as a list of their coefficients) for a given

simple graph: the Ising partition function, the q-state Potts partition function ($q \neq 0, 1, 2$), the reliability polynomial, the chromatic polynomial, and the flow polynomial. Moreover, we have $\exp(\Omega(n))$ lower bounds for the following counting problems on multigraphs: # perfect matchings, # cycle covers in digraphs, # connected spanning subgraphs, all-terminal graph reliability with given edge failure probability $p > 0$, # nowhere-zero k-flows ($k \neq 0, \pm 1$), and # acyclic orientations.

The lower bound for counting the number of perfect matchings holds even in bipartite graphs, where an $O(1.414^n)$ algorithm is given by Ryser's formula. Such algorithms are also known for general graphs [3], the current best bound is $O(1.619^n)$ [14].

For simple graphs, we have $\exp(\Omega(m/\log m))$ lower bounds for # perfect matchings and # cycle covers in digraphs.

3 Hardness of the Permanent

This section contains the proof of Thm. 2. With $[0, n] = \{0, 1, \ldots, n\}$ we establish the reduction chain #3-Sat $\preccurlyeq$ Perm$^{-1,0,1}$ $\preccurlyeq$ Perm$^{[0,n]}$ $\preccurlyeq$ Perm^{01} while taking care of the instance sizes.

Proof (of Thm. 2). First, to prove (i), we reduce #3-Sat in polynomial time to Perm$^{-1,0,1}$ such that 3-CNF formulas φ with m clauses are mapped to graphs G with $O(m)$ edges. For technical reasons, we preprocess φ such that every variable x occurs equally often as a positive literal and as a negative literal $\bar{x}$ (e.g., by adding trivial clauses of the form $(x \vee \bar{x} \vee \bar{x})$ to φ). We construct G with $O(m)$ edges and weights $w : E \to \{\pm 1\}$ such that #Sat(φ) can be derived from $\operatorname{per} G$ in polynomial time. For weighted graphs, the permanent is

$$\operatorname{per} G = \sum_{C \subseteq E} w(C)\,, \qquad \text{where } w(C) = \prod_{e \in C} w(e)\,.$$

The sum above is over all cycle covers C of G, that is, subgraphs (V, C) with an in- and outdegree of 1 at every vertex.

In Fig. 2, the gadgets of the construction are depicted. For every variable x that occurs in φ, we add a *selector gadget* to G. For every clause $c = \ell_1 \vee \ell_2 \vee \ell_3$ of φ, we add a *clause gadget* to G. Finally, we connect the edge labelled by a literal ℓ in the selector gadget with all occurrences of ℓ in the clause gadgets, using *equality gadgets*. This concludes the construction of G.

The number of edges of the resulting graph G is linear in the number of clauses. The correctness of the reduction follows along the lines of [17] and [5]. The satisfying assignments stand in bijection to cycle covers of weight $(-1)^i 2^j$ where i (resp. j) is the number of occurrences of literals set to false (resp. true) by the assignment, and all other cycle covers sum up to 0. Since we preprocessed φ such that $i = j$, we obtain $\operatorname{per} G = (-2)^i \cdot \#\text{Sat}(\varphi)$.

To prove (ii), we reduce Perm$^{-1,0,1}$ in polynomial time to Perm$^{[0,n]}$ by interpolation: On input G, we conceptually replace all occurrences of the weight -1

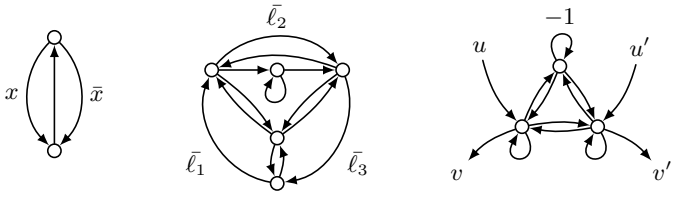


Fig. 2. Left: A selector gadget for variable x. Depending on which of the two cycles is chosen, we assume x to be set to true or false. Middle: A clause gadget for the clause $\ell_1 \vee \ell_2 \vee \ell_3$. The gadget allows all possible configurations for the outer edges, except for the case that all three are chosen (which would correspond to $\ell_1 = \ell_2 = \ell_3 = 0$). Right: An equality gadget that replaces two edges uv and $u'v'$. The top loop carries a weight of -1. It can be checked that the gadget contributes a weight of -1 if all four outer edges are taken, $+2$ if none of them is taken, and 0 otherwise.

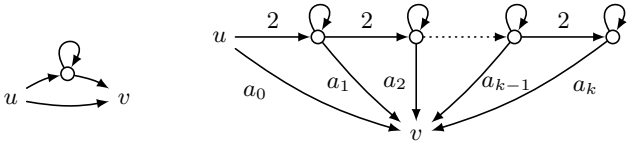


Fig. 3. Left: This gadget simulates in unweighted graphs edges uv of weight 2. Right: This gadget simulates edges uv of weight $a = \sum_{i=0}^{k} a_i 2^i$ with $a_i \in \{0, 1\}$.

by a variable x and call this new graph G_x. We can assume that only loops have weight x in G_x because the output graph G from the previous reduction has weight -1 only on loops. Then $p(x) = \operatorname{per} G_x$ is a polynomial of degree $d \leq n$.

If we replace x by a value $a \in [0, n]$, then G_a is a weighted graph with as many edges as G. As a consequence, we can use the oracle to compute $\operatorname{per} G_a$ for $a = 0, \ldots, d$ and then interpolate, to get the coefficients of the polynomial $p(x)$. At last, we return the value $p(-1) = \operatorname{per} G$. This completes the reduction, which queries the oracle $d + 1$ graphs that have at most m edges each.

For part (iii), we have to get rid of positive weights. Let G_a be one query of the last reduction. Again we assume that $a \leq n$ and that weights $\neq 1$ are only allowed at loop edges. We replace every edge of weight a by the gadget that is drawn in Fig. 3, and call this new unweighted graph G'. It can be checked easily that the gadget indeed simulates a weight of a (parallel paths correspond to addition, serial edges to multiplication), i.e., $\operatorname{per} G' = \operatorname{per} G_a$. Unfortunately, the reduction blows up the number of edges by a superconstant factor: The number of edges of G' is $m(G') \leq (m + n \log a) \leq O(m + n \log n)$. But since $m(G')/\log m(G') \leq O(m)$, the reduction shows that (iii) follows from (ii). ∎

These results immediately transfer to counting the number of perfect matchings in a graph even if the graph is restricted to be bipartite.

4 Hyperbolas in the Tutte Plane

Our first goal will be to show that the Tutte polynomial is hard "for all hyperbolas" $(x-1)(y-1)=q$, except for $q=0$ (which we understand only partially), $q=1$ (which is in P), and for $q=2$ (which he handle separately in the full paper by a reduction from the permanent). From the hyperbolas, we will specialise the hardness result to individual points in the following sections.

4.1 The Multivariate Tutte Polynomial

We need Sokal's multivariate version of the Tutte polynomial, defined in [22] as follows. Let $G=(V,E)$ be an undirected graph whose edge weights are given by a function $\mathbf{w}\colon E\to\mathbb{Q}$. Then

$$Z(G;q,\mathbf{w})=\sum_{A\subseteq E} q^{k(A)}\prod_{e\in A}\mathbf{w}(e)\,, \tag{3}$$

where $k(A)$ is the number of connected components in the subgraph (V,A). If $\mathbf{w}$ is single-valued, in the sense that $\mathbf{w}(e)=w$ for all $e\in E$, we slightly abuse notation and write $Z(G;q,w)$. With a single-valued weight function, the multivariate Tutte polynomial essentially equals the Tutte polynomial,

$$\begin{aligned} T(G;x,y)&=(x-1)^{-k(E)}(y-1)^{-|V|}Z(G;q,w)\,,\\ &\text{where } q=(x-1)(y-1) \text{ and } w=y-1\,, \end{aligned} \tag{4}$$

see [22, eq. (2.26)]. The conceptual strength of the multivariate perspective is that it turns the Tutte polynomial's second variable y, suitably transformed, into an edge weight of the input graph. In particular, the multivariate formulation allows the graph to have different weights on different edges, which turns out to be a dramatic technical simplification even when, as in the present work, we are ultimately interested in the single-valued case.

Sokal's polynomial vanishes at $q=0$, so we will sometimes work with the polynomial

$$Z_0(G;q,\mathbf{w})=\sum_{A\subseteq E} q^{k(A)-k(E)}\prod_{e\in A}\mathbf{w}(e)\,,$$

which gives something non-trivial for $q=0$ and is otherwise a proxy for Z:

$$Z(G;q,\mathbf{w})=q^{k(E)}Z_0(G;q,\mathbf{w})\,. \tag{5}$$

4.2 Three-Terminal Minimum Cut

We first establish that with two different edge weights, one of them negative, the multivariate Tutte polynomial computes the size of a 3-terminal minimum cut, for which we observe hardness under #ETH in the full paper. This connection has been used already in [8,9], with different reductions, to prove hardness of approximation.

The graphs we will look at are connected and have rather simple weight functions. The edges are partitioned into two sets $E\dot\cup T$, and for fixed rational w the weight function is given by

$$\mathbf{w}(e) = \begin{cases} -1, & \text{if } e \in T, \\ w, & \text{if } e \in E. \end{cases} \tag{6}$$

For such a graph, we have

$$Z_0(G; q, \mathbf{w}) = \sum_{A \subseteq E \cup T} q^{k(A)-1} w^{|A \cap E|} (-1)^{|A \cap T|}. \tag{7}$$

For fixed G and q, this is a polynomial in w of degree at most m.

Lemma 1. *Let q be a rational number with $q \notin \{1, 2\}$. Computing the coefficients of the polynomial $w \mapsto Z_0(G; q, \mathbf{w})$, with $\mathbf{w}$ as in (6), for a given simple graph G requires time $\exp(\Omega(m))$ under #ETH.*

Moreover, this is true even if $|T| = 3$.

From this result, the argument continues in two directions. For simple graphs and certain parts of the Tutte plane, we proceed in §4.3 and §5. For *nonsimple* graphs and certain (other) parts of the Tutte plane, we can use just thickening and interpolation, which we lay out in the full paper.

4.3 The Tutte Polynomial Along a Hyperbola

To apply Lemma 1 to the Tutte polynomial, we need to get rid of the negative edges, so that the weight function is single-valued. In [9], this is done by thickenings and stretches, which we need to avoid. However, since the number of negative edges is small (in fact, 3), we can use another tool, deletion–contraction. We will omit the case $q = 0$ from this analysis, because we won't need it later, so we can work with Z instead of Z_0.

A *deletion–contraction* identity expresses a function of the graph G in terms of two graphs $G - e$ and G/e, where

$G - e$ arises from G by *deleting* the edge e and
G/e arises from G by *contracting* the edge e, that is, deleting it and identifying its endpoints so that remaining edges between these two endpoints become loops.

It is known [22, eq. (4.6)] that $Z(G; q, \mathbf{w}) = Z(G - e; q, \mathbf{w}) + \mathbf{w}(e) Z(G/e; q, \mathbf{w})$.

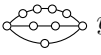
Lemma 2. *Computing the coefficients of the polynomial $v \mapsto Z(G; q, v)$ for a given simple graph G requires time $\exp(\Omega(m))$ under #ETH, for all $q \notin \{0, 1, 2\}$.*

5 Generalised Theta Graphs

We now prove Thm. 3 (ii) by showing that most points (x, y) of the Tutte plane, are as hard as the entire hyperbola on which they lie, even for sparse, simple graphs. The drawback of our method is that we loose a polylogarithmic factor in the exponent of the lower bound and we do not get any results if

$q := (x-1)(y-1) \in \{0,1,2\}$ or if $x \in \{-1,0\}$. However, the results are particularly interesting for the points on the line $y = -1$, for which we know no other good exponential lower bounds under #ETH, even in more general graph classes. We remark that the points $(-1,-1)$, $(0,-1)$, and $(\frac{1}{2},-1)$ on this line are known to admit a polynomial-time algorithm, and indeed our hardness result does not apply here. Also, since our technique does not work in the case $q = 0$, the point $(1,-1)$ remains mysterious.

For a set $S = \{s_1, \ldots, s_k\}$ of positive integers, the *generalised Theta graph* Θ_S consists of two vertices x and y joined by k internally disjoint paths of $s_1, \ldots, s_k$ edges, respectively. For example,

$\Theta_{\{2,3,5\}}$ is  .

For such a graph Θ_S, we will study the behaviour of the tensor product $G \otimes \Theta_S$ defined by Brylawski [6] as follows: given $G = (V, E)$, replace every edge $xy \in E$ by (a fresh copy of) Θ_S. What makes the $\otimes$-operation so useful in the study of Tutte polynomials is that the Tutte polynomial of $G \otimes H$ can be expressed in terms of the Tutte polynomials of G and H, as studied by Sokal. The necessary formulas for $Z(G \otimes \Theta_S)$ can be derived from [21, prop 2.2, prop 2.3]. We present them here for the special case where all edge weights are the same.

Lemma 3 (Sokal). *Let q and w be rational numbers with $w \neq 0$ and $q \notin \{0, -2w\}$. Then, for all graphs G and finite sets S of positive integers,*

$$Z(G \otimes \Theta_S; q, w) = q^{|E|-|S|} \cdot \prod_{s \in S} \left((q+w)^s - w^s\right)^{|E|} \cdot Z(G; q, w_S)\,, \tag{8}$$

where

$$w_S = -1 + \prod_{s \in S} \left(1 + \frac{q}{(1+q/w)^s - 1}\right)\,. \tag{9}$$

Our plan is to compute the coefficients of the monovariate polynomial $w \mapsto Z(G; q, w)$ for given G and q by interpolation from sufficiently many evaluations of $Z(G; q, w_S) \sim Z(G \otimes \Theta_S; q, w)$. For this, we need that the number of different w_S is at least $|E| + 1$, one more than the degree of the polynomial.

Lemma 4. *Let q and w be rational numbers with $w \neq 0$ and $q \notin \{0, -w, -2w\}$. For all integers $m \geq 1$, there exist sets $S_0, \ldots, S_m$ of positive integers such that*

(i) $\sum_{s \in S_i} s \leq O(\log^3 m)$ for all i, and
(ii) $w_{S_i} \neq w_{S_j}$ for all $i \neq j$.

Furthermore, the sets S_i can be computed in time polynomial in m.

This lemma together with interpolation establishes Thm. 3 (ii).

Acknowledgements. The authors are grateful to Leslie Ann Goldberg and Andreas Björklund for valuable comments.

References

1. Agrawal, M.: Determinant versus permanent. In: Proceedings of the 25th International Congress of Mathematicians, ICM, vol. 3, pp. 985–997 (2006)
2. Berkowitz, S.J.: On computing the determinant in small parallel time using a small number of processors. Information Processing Letters 18(3), 147–150 (1984)
3. Björklund, A., Husfeldt, T.: Exact algorithms for exact satisfiability and number of perfect matchings. Algorithmica 52(2), 226–249 (2008)
4. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Computing the Tutte polynomial in vertex-exponential time. In: FOCS, pp. 677–686 (2008)
5. Bläser, M., Dell, H.: Complexity of the cover polynomial. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 801–812. Springer, Heidelberg (2007)
6. Brylawski, T.: The Tutte polynomial, Matroid theory and its applications. In: Centro Internazionale Matematico Estivo, pp. 125–275 (1982)
7. Giménez, O., Hliněný, P., Noy, M.: Computing the Tutte polynomial on graphs of bounded clique-width. SIAM J. on Discrete Mathematics 20, 932–946 (2006)
8. Goldberg, L.A., Jerrum, M.: The complexity of ferromagnetic Ising with local fields. Combinatorics, Probability and Computing 16(1), 43–61 (2007)
9. Goldberg, L.A., Jerrum, M.: Inapproximability of the Tutte polynomial. Information and Computation 206(7), 908–929 (2008)
10. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? Journal of Computer and System Sciences 63(4), 512–530 (2001)
11. Istrail, S.: Statistical mechanics, three-dimensionality and NP-completeness. I. universality of intractability for the partition function of the Ising model across nonplanar lattices. In: STOC, pp. 87–96 (2000)
12. Jaeger, F., Vertigan, D.L., Welsh, D.J.: On the computational complexity of the Jones and Tutte polynomials. Math. Proc. Cambridge 108(1), 35–53 (1990)
13. Jerrum, M., Snir, M.: Some exact complexity results for straight-line computations over semirings. Journal of the ACM 29(3), 874–897 (1982)
14. Koivisto, M.: Partitioning into sets of bounded cardinality. In: IWPEC, pp. 258–263 (2009)
15. Kutzkov, K.: New upper bound for the #3-SAT problem. Information Processing Letters 105(1), 1–5 (2007)
16. Lawler, E.L.: A note on the complexity of the chromatic number problem. Information Processing Letters 5, 66–67 (1976)
17. Papadimitriou, C.M.: Computational Complexity. Addison-Wesley, Reading (1994)
18. Raz, R.: Multi-linear formulas for permanent and determinant are of superpolynomial size. Journal of the ACM 56(2), 1–17 (2009)
19. Ryser, H.J.: Combinatorial mathematics. Carus Math. Monographs, vol. 14. Mathematical Association of America (1963)
20. Sekine, K., Imai, H., Tani, S.: Computing the Tutte polynomial of a graph of moderate size. In: Staples, J., Katoh, N., Eades, P., Moffat, A. (eds.) ISAAC 1995. LNCS, vol. 1004, pp. 224–233. Springer, Heidelberg (1995)
21. Sokal, A.D.: Chromatic roots are dense in the whole complex plane. Combinatorics, Probability and Computing 13(02), 221–261 (2004)
22. Sokal, A.D.: The multivariate Tutte polynomial (alias Potts model) for graphs and matroids. In: Surveys in Combinatorics. London Mathematical Society Lecture Note Series, vol. 327, pp. 173–226. Cambridge University Press, Cambridge (2005)
23. Valiant, L.G.: The complexity of computing the permanent. Theoretical Computer Science 8(2), 189–201 (1979)

On Approximate Horn Formula Minimization

Amitava Bhattacharya[1], Bhaskar DasGupta[2,⋆], Dhruv Mubayi[3,⋆⋆], and György Turán[3,4,⋆⋆⋆]

[1] School of Mathematics, Tata Institute of Fundamental Research, Colaba, Mumbai 400 005, India
amitava@math.tifr.res.in
[2] Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607-7053
dasgupta@cs.uic.edu
[3] Department of Mathematics, Statistics & Computer Science, University of Illinois at Chicago, Chicago, IL 60607-7045
mubayi@math.uic.edu, gyt@uic.edu
[4] Research Group on Artificial Intelligence, University of Szeged, Hungary

Abstract. The minimization problem for Horn formulas is to find a Horn formula equivalent to a given Horn formula, using a minimum number of clauses. A $2^{\log^{1-\epsilon}(n)}$-inapproximability result is proven, which is the first inapproximability result for this problem. We also consider several other versions of Horn minimization. The more general version which allows for the introduction of new variables is known to be too difficult as its equivalence problem is co-NP-complete. Therefore, we propose a variant called *Steiner-minimization*, which allows for the introduction of new variables in a restricted manner. Steiner-minimization of Horn formulas is shown to be MAX-SNP-hard. In the positive direction, a $o(n)$, namely, $O(n \log\log n/(\log n)^{1/4})$-approximation algorithm is given for the Steiner-minimization of definite Horn formulas. The algorithm is based on a new result in algorithmic extremal graph theory, on partitioning bipartite graphs into complete bipartite graphs, which may be of independent interest. Inapproximability results and approximation algorithms are also given for restricted versions of Horn minimization, where only clauses present in the original formula may be used.

1 Introduction

The CNF minimization problem is to find a shortest CNF expression equivalent to a given expression. This problem has been studied in different versions for many decades in switching theory, computer science and engineering, and it is still a topic of active research, both in complexity theory and circuit design. Umans [40,41] showed Σ_p^2-completeness and a $O(n^{1-\varepsilon})$-inapproximability result

⋆ Supported by NSF Grant IIS-0346973 and DIMACS special focus on Computational and Mathematical Epidemiology.
⋆⋆ Supported by NSF grant DMS-0653946.
⋆⋆⋆ Supported by NSF grant CCF-0916708.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 438–450, 2010.

for this problem. Horn minimization is the special case of CNF minimization for Horn formulas. Horn formulas are conjunctions of *Horn clauses*, *i.e.*, of disjunctions containing *at most one* unnegated variable. Horn clauses can also be written as implications. For instance, $\bar{a} \vee \bar{b} \vee c$ is a Horn clause which can also be written as $a, b \rightarrow c$.

Horn formulas are an expressive and tractable fragment of propositional logic, and therefore provide a basic framework for knowledge representation and reasoning [35]. Horn formulas are, for example, a natural framework for representing systems of rules for expert systems. An interesting potential new application area for Horn formulas is the automated, interactive development of large-scale knowledge bases of commonsense knowledge (see [36] for the description of such a project). This application has algorithmic aspects involving knowledge representation, reasoning, learning and knowledge update. A model incorporating these aspects, called *Knowledge Base Learning (KnowBLe)* is formulated in [29] (see also [28,30]) for related work). Efficient algorithms for approximate Horn minimization would be useful in these applications.

Satisfiability of Horn formulas can be decided in linear time and the equivalence of Horn formulas can be decided in polynomial time [25]. Thus Horn minimization is expected to be easier than CNF minimization. Horn minimization was shown to be NP-complete by Hammer and Kogan [21] if the number of literals is to be minimized, and by Ausiello *et al.* [4] and Boros and Čepek [8] if the number of clauses is to be minimized. On the positive side, Hammer and Kogan [22] gave a polynomial algorithm for minimizing quasi-acyclic Horn formulas, which include both acyclic and 2-Horn formulas. It was also shown in [21] that there is an efficient $(n-1)$-approximation algorithm for general Horn minimization, where n is the *number of different variables* in the formula (not the number of variable occurrences). As noted in [18], such an algorithm is also provided by the Horn formula learning algorithm of [3].

1.1 Contributions of This Paper

First, in Theorem 3 we prove a $2^{\log^{1-\epsilon}(n)}$-inapproximability result for Horn minimization assuming NP$\not\subseteq$DTIME$(n^{\text{polylog(n)}})$ via a reduction from the MINREP problem [26]. This seems to be the first inapproximability result for this problem. We next consider several other versions of the Horn minimization problem. Depending on the application, different versions may be relevant and thus of interest for exploration of their approximability properties.

It may be possible to *add new variables* in order to compress the formula. For example, the formula

$$\varphi = \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} (x_i \rightarrow y_j) \tag{1}$$

having n^2 clauses can be compressed to the $2n$ clause formula

$$\psi = \bigwedge_{i=1}^{n} (x_i \rightarrow z) \wedge \bigwedge_{j=1}^{n} (z \rightarrow y_j), \tag{2}$$

where z is a new variable. Note that φ and ψ are clearly *not* equivalent, *e.g.*, φ does not depend on z, while ψ does. On the other hand, φ and ψ *are* equivalent in the sense that they both imply the same set of clauses over the original variables $x_1, \ldots, x_n, y_1, \ldots, y_n$. Thus, in terms of the knowledge base application, the new variable z can be thought of as being *internal* to the knowledge base and invisible to the user. Flögel *et al.* [17] showed that deciding the equivalence of such extended Horn formulas is co-NP-complete. This is bad news as it shows that the extended version is too expressive and therefore intractable[1].

On the other hand, notice that in the example above the new variable is added in a rather *restricted manner*. Formula (1) can be thought of as a complete directed bipartite graph with parts $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$. Formula (2), then, represents the operation of adding a new node z in the middle, with edges from the x_i's to z and from z to the y_j's. The two graphs have the same reachability relations as far as the original nodes are concerned. Using the similarity to Steiner problems where new points may be added [23], we refer to this operation as a *Steiner extension* (a formal definition appears in Section 5). As we observe, in contrast to general extensions, the equivalence of Steiner extensions *can* be decided efficiently (Corollary 1). Thus this type of extension could be considered as a tractable alternative in the applications mentioned. The *Steiner minimization* problem for Horn formulas is then to find an equivalent Steiner-extended Horn formula, for a given Horn formula, with a minimum number of clauses. We show in Theorem 5 that this problem is MAX-SNP-hard. On the other hand, in Theorem 6 we prove that there is an efficient $O(n \log\log n/(\log n)^{1/4})$-approximation algorithm for this problem, where n is the number of variables in the original formula. This is the first approximation algorithm for Horn minimization with a $o(n)$ approximation guarantee.

The algorithm for Steiner minimization makes use of an algorithmic result on the *partition of bipartite graphs into complete bipartite graphs (*i.e., *bicliques)*, which may be of interest on its own. It was shown by Chung, Erdős and Spencer [13] and Bublitz [10] that the edges of every n-vertex graph can be partitioned into complete bipartite graphs such that the sum of the number of vertices in these bipartite graphs[2] is $O(n^2/\log n)$, and this is asymptotically the best possible. Tuza [39] gave an analogous result for bipartite graphs. These results are based on a counting argument due to Kővári, Sós and Turán [27], which shows that sufficiently dense graphs contain large complete bipartite subgraphs, and thus are non-constructive. Kirchner [24] considered the problem of finding an algorithmic version, and gave an efficient algorithm to find complete balanced bipartite graphs of size $\Omega(\sqrt{\log n})$ in dense graphs. In a previous paper [33] we improved this to the optimal $\Omega(\log n)$,

[1] Introducing new variables has been considered earlier, going back to Tseitin [38]. While there are many exponential lower bounds for resolution proofs (see, *e.g.*, [14]), complexity-theoretic results suggest that proving such results for extended resolution is much harder [15].

[2] Note that the complexity of a partition is not measured by the number of graphs in it, but by a different measure, which comes from circuit complexity [37].

and as a corollary, showed that partitions proved to *exist* in [10,13] can also be *found efficiently*[3].

In this paper we give an algorithmic version for the bipartite case. We show in Theorem 1 that the edges of every bipartite graph with sides a and b, where $a \geq b$, can be partitioned into complete bipartite graphs such that the sum of the number of vertices of these graphs is $O((ab/\log a) + a\log b + a)$, and we give an efficient algorithm to find such a partition.

We also consider *restricted* versions of the Horn minimization problem, where one is restricted to use clauses from the original formula. Such a restriction may be justified in applications where the particular rules, provided by an expert, are supposed to be meaningful and thus cannot be replaced. The goal is to eliminate redundant rules. Modifying the construction of Theorem 3, in Theorem 7 we prove $2^{\log^{1-\epsilon}(n)}$-inapproximability for the restricted case, which holds even if the input formula has clauses of size at most 3.

One may want to optimize a Horn formula in the restricted sense either by *minimizing the number of rules left*, or by *maximizing the number of rules removed*. The two versions may differ in their approximability (cf. the maximum independent set and the minimum vertex cover problems for graphs). As (1) suggests, Horn formulas with clauses of the form $x \to y$ correspond to directed graphs. For such formulas, optimization corresponds to *transitive reduction* problems for directed graphs. Thus approximation algorithms for these directed graph problems (in both versions) may be applied for Horn formulas. Examples of this connection are given in Theorem 8.

The rest of the paper is organized as follows. Bipartite graph decompositions are discussed in Section 3. Section 4 contains the inapproximability result for Horn minimization. Horn minimization with new variables is discussed in Section 5, and restricted Horn minimization in Section 6.

2 Preliminaries

A *clause* is a disjunction of literals. A *Horn clause* is a clause with at most one unnegated variable. A *definite* Horn clause has exactly one unnegated variable, called its *head*; the other variables form its *body*. A *negative* clause consists of only negated variables. The *size* of a clause is the number of its literals. A clause of size 1 (resp., 2) is a *unit* (resp., *binary*) clause. A *(definite) Horn formula* is a conjunction of (definite) Horn clauses. The *size* of a formula is the number of its clauses. A *k-Horn formula* is a Horn formula with clauses of size at most k.

A clause C is an *implicate* of a formula φ (also written as $\varphi \models C$) if every truth assignment satisfying φ also satisfies C. An implicate is a *prime* implicate

[3] Extremal combinatorics provides results on the existence of substructures. The results of [24] and [33] can be viewed as *algorithmic extremal combinatorics* as they also give efficient algorithms to actually find such substructures. Previous results in this direction are given in [2]. The results of [2] apply to dense graphs and find substructures of constant size, while here we have to handle sparser graphs as well and to find substructures of nonconstant size.

if none of its proper subclauses is an implicate. The *resolution* operation takes two clauses of the form $C_1 \vee x$ and $C_2 \vee \bar{x}$ and produces the clause $C_1 \vee C_2$. For basic properties of resolution, see, e.g. [25].

Deciding whether a *definite Horn* clause C is an implicate of a *definite Horn* formula φ can be decided by a simple and well-known marking procedure often called *forward chaining*. The procedure begins by marking the variables in the body of C. If every variable in the body of a clause in φ is marked then its head is marked as well. This is repeated as long as new variables get marked. Then it holds that C is an implicate of φ iff its head gets marked.

3 Partitioning/Covering Bipartite Graphs Using Bicliques

Let $G = G(A, B, E)$ be a bipartite graph with parts A, B of sizes a, b, respectively, and edge set E of size m. We assume w.l.o.g. that $a \geq b$. A bipartite graph is *balanced* if $|A| = |B|$. The complete bipartite graph (or biclique) with parts of size p and q is denoted by $K_{p,q}$. We consider bicliques $G_i = (A_i, B_i, E_i)$ for $i = 1, \ldots, t$ such that $A_i \subseteq A, B_i \subseteq B$, and $(E_1, \ldots, E_t)$ is a partition, resp. a cover, of E. The cost of such a decomposition is $\sum_{i=1}^{t}(|A_i| + |B_i|)$. The problem is to find a decomposition of small cost. The trivial decomposition into single edges has a cost of $2m \leq 2ab$.

We consider two versions of the problem. In the first version we are interested in finding a partition such that its size is upper bounded by some function of a and b, independent of m.

Theorem 1. *For every bipartite graph G one can find a partition of cost $O\left(\frac{ab}{\log a} + a \log b + a\right)$ in polynomial time.*

The decomposition is found by iteratively finding large bipartite subgraphs. There are two procedures, depending on a carefully chosen notion of density. Let $3 \leq b \leq a$, $6a \leq m \leq ab$ and $f(a, b, m) = \left\lfloor \frac{\log a}{\log(2eab/m)} \right\rfloor$; note that the case $b \leq 2$ is trivial.

Lemma 1. *Suppose that $m \geq af(a, b, m)$. Then there is a polynomial time algorithm that finds a $K_{q,q}$ in G with $q = f(a, b, m)$.*

Lemma 2. *Suppose that $m < af(a, b, m)$. Then there is a polynomial time algorithm that finds a $K_{q,q}$ in G with $q = \lfloor m/a \rfloor$.*

Remark 1. If G is a star then $b = 1$ and the optimal decomposition has cost $a + 1$, hence the upper bound $ab/\log a$ claimed in [39] does not hold, and an additional term (or some other modification) is needed. It is open whether the quantity $a \log b + a$ can be improved.

In the second version we are interested in finding a partition (resp., cover) of minimal cost. For technical reasons, we use a slightly different cost function here (using this cost function would not change anything in the previous result). The size

of $K_{p,q}$ is $p{\cdot}q$ and the modified cost $\text{cost}'(K_{p,q})$ of $K_{p,q}$ is $p{\cdot}q$ if $p = 1$ or $q = 1$, and is $p+q$ otherwise. The reason for using the modified cost measure for Horn minimization is that when a set of Horn clauses $\wedge_{i=1}^{p} \wedge_{j=1}^{q} (x_i \rightarrow y_j)$ corresponding to a biclique $K_{p,q}$ is replaced by a set of Horn clauses $\wedge_{i=1}^{p}(x_i \rightarrow z) \bigwedge \wedge_{j=1}^{q}(z \rightarrow y_j)$ by introducing a new variable z if $p, q > 1$, and is left unchanged otherwise, the size of the new formula is $\text{cost}'(p, q)$. We define the Linear-Cost-Biclique-Cover (resp., Linear-Cost-Biclique-Partition) problem as follows: given a bipartite graph $G = (A, B, E)$, cover (resp., partition) its edges with bicliques of minimum total *modified* cost. The minimization of the number of bicliques in a cover was shown to be NP-complete by Orlin [34]. The following result follows by an approximation-preserving reduction from the maximum independent set problem for 3-regular graphs.

Theorem 2. *Assuming* P≠NP*,* Linear-Cost-Biclique-Cover *and* Linear-Cost-Biclique-Partition *cannot be approximated in polynomial time within an approximation ratio of* $1 + (1/1138)$ *even if the input graph has no biclique of size more than* 6*.*

4 Inapproximability

Theorem 3. *For any fixed* $0 < \epsilon < 1$*, unless* NP⊆DTIME$(n^{polylog(n)})$*, the Horn minimization problem for definite Horn formulas is* $2^{\log^{1-\epsilon} n}$*-inapproximable.*

The reduction is from the Minrep problem [26]. An instance M is given by a bipartite graph $G = (A, B, E)$ with $|E| = m$, a partition of A into equal-size subsets $A_1, A_2, \ldots, A_\alpha$ and a partition of B into equal-size subsets $B_1, B_2, \ldots, B_\beta$. One can define a natural bipartite super-graph H in the following manner. H has a super-vertex for every A_i and B_j. There is a super-edge between A_i and B_j if and only if there exists $u \in A_i$ and $v \in B_j$ such that (u, v) is an edge of G. Let the number of super-edges be p. A pair of nodes u and v *witnesses* a super-edge (A_i, B_j) provided $u \in A_i$, $v \in B_j$ and the edge (u, v) exists in G. A set of nodes S of G witnesses a super-edge if and only if there exists at least one pair of nodes in S that witnesses the super-edge. The goal of Minrep is to find $A' \subseteq A$ and $B' \subseteq B$ such that $A' \cup B'$ witnesses every super-edge of H and $|A'| + |B'|$ is as small as possible. The *size* of an optimal solution is denoted by $OPT(M)$. Let $s = |A| + |B|$. It is shown in Kortsarz *et al.* [26] that Minrep is $2^{\log^{1-\varepsilon} n}$-inapproximable under the complexity-theoretic assumption NP⊈DTIME$(n^{\text{polylog(n)}})$.

Consider an instance M of Minrep . Let t be a sufficiently large positive integer to be fixed later. We construct a definite Horn formula φ. For simplicity and with an abuse of notation, some variables in φ are denoted as the corresponding objects (vertices and super-edges) in the Minrep instance. The formula φ contains *amplification* variables $x_1, \ldots, x_t$, *node* variables u for every vertex u in $A \cup B$ and *super-edge variables* e for every super-edge e in H. The clauses of φ belong to the following groups:

amplification clauses: there is a clause $x_i \to u$ for every $i \in \{1, \ldots, t\}$ and for every $u \in A \cup B$,

witness clauses: there is a clause $u, v \to e$ for every super-edge e of H and for every pair of nodes $u \in A$ and $v \in B$ witnessing e,

feedback clauses: there is a clause $e_1, \ldots, e_p \to u$ for every $u \in A \cup B$, where $e_1, e_2, \ldots, e_p$ are the super-edges of H.

As φ is definite, all its prime implicates are definite. Also, as φ consists of non-unit definite clauses, all its prime implicates are non-unit (the all-zero vector satisfies φ and falsifies all unnegated variables). For a further analysis of the prime implicates of φ, we make use of forward chaining.

Lemma 3. *Let x be an amplification variable. Then the prime implicates containing x are clauses of the form $x \to v$, where v is a node or super-edge variable.*

Lemma 4. *Let U be a set of node variables such that U is not a solution to* MINREP *, and U' be the set of super-edge variables witnessed by U. Then every implicate with body contained in $U \cup U'$ has head in U'.*

Lemma 5. *Let x be an amplification variable and let ψ be a prime and irredundant Horn formula equivalent to φ. Then ψ has at least $OPT(M)/2$ clauses containing x.*

Based on these lemmas one can prove that the reduction is gap-preserving.

Lemma 6. (Gap preserving reduction lemma)

(a) *If $OPT(M) = \alpha + \beta$, then $OPT(\varphi) \leq t \cdot (\alpha + \beta) + m + s$.*
(b) *If $OPT(M) \geq (\alpha + \beta) \cdot 2^{\log^{1-\varepsilon} s}$ then, $OPT(\varphi) \geq t(\alpha + \beta) \cdot 2^{\log^{1-\varepsilon} s}/2$.*

Our result now follows from the inapproximability result for MINREP mentioned above.

5 Formulas with New Variables

In this section we consider versions of the Horn minimization problem where one can introduce new variables in order to compress the formula.

5.1 General Extensions

First we consider the general version where there is no restriction on the way new variables are introduced.

Definition 1 (Generalized equivalence). [17] *Let X be a set of variables. Formulas φ and ψ are X-equivalent if for every clause C involving only variables from X it holds that $\varphi \models C$ iff $\psi \models C$.*

Consider the set of variables $X = \{x_1, \ldots, x_n, y_1, \ldots, y_n, u\}$ and the 2^n-clause Horn formula $\varphi = \bigwedge(v_1, \ldots, v_n \rightarrow u)$ where $v_i \in \{x_i, y_i\}$ for $i = 1, \ldots, n$, and the conjunction includes all possible such selections. As no resolutions can be performed, it follows that all the prime implicates of φ are the clauses themselves. Let now $\{z_1, \ldots, z_n\}$ be new variables. Then the $(2n+1)$-clause Horn formula $\psi = (z_1, \ldots, z_n \rightarrow u) \wedge \bigwedge_{i=1}^{n} (x_i \rightarrow z_i) \wedge (y_i \rightarrow z_i)$ is X-equivalent to φ. Thus the introduction of new variables can lead to an exponential compression in size.

For knowledge representation formalisms it is useful to have an efficient procedure to decide equivalence. Thus the following result of [17] suggests that general extensions of Horn formulas are too general for applications.

Theorem 4. [17] *Generalized equivalence of definite Horn formulas is* co-NP-complete.

5.2 Steiner Extension

The proof of Theorem 4 shows that generalized equivalence is already hard if new variables are introduced in a rather restricted manner. This gives a motivation to consider even more stringent restrictions on the introduction of new variables.

Definition 2 (Steiner extension). *Let φ be a Horn formula and X be a subset of its variables. Then φ is a* Steiner extension *over X if every variable* not *in X occurs in φ either as a head, or as a single body variable in a binary definite clause having its head in X.*

The corresponding notion of equivalence is the following.

Definition 3 (Steiner equivalence). *Let X be a set of variables. Horn formulas φ and ψ are* Steiner *X-equivalent if*

- *φ and ψ are X-equivalent,*
- *both φ and ψ are Steiner extensions over X.*

The Horn formulas in (1) and (2) in Section 1.1 are both Steiner extensions over $X = \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$, and they are Steiner X-equivalent. On the other hand, the example in Section 5.1 is not a Steiner extension as additional variables occur in the body of a non-binary clause. In contrast to Theorem 4, Steiner equivalence can be decided efficiently.

Proposition 1. *There is a polynomial algorithm which, given a Steiner extension φ over X, computes a Steiner X-equivalent Horn formula $\psi(X)$ containing only the variables in X such that $size(\psi) = O(size(\varphi)^2)$.*

The Steiner X-equivalence of φ_1 and φ_2 can be decided by using Proposition 1 to produce formulas $\psi_1(X)$ and $\psi_2(X)$ and checking their equivalence.

Corollary 1. *Steiner equivalence of Horn formulas is in P.*

The minimization problem for Steiner equivalence is the following.

Definition 4 (Steiner minimization of Horn formulas). *Given a Horn formula φ over variables X, find a minimal size Horn formula that is Steiner X-equivalent to φ.*

Using the correspondence between bipartite graphs and Horn formulas of binary clauses discussed earlier, Theorem 2 can be used to show the following.

Theorem 5. *Steiner minimization of definite Horn formulas is* MAX-SNP-hard.

We now show that Steiner minimization of definite Horn formulas has an efficient approximation algorithm with performance guarantee $o(n)$.

Remark 2. It may be assumed w.l.o.g. that Horn formulas to be minimized have no unit clause prime implicates. This holds as every prime representation can be partitioned into those unit clauses and a set of clauses not containing any variable that occurs in a unit clause prime implicate. The second half then can be minimized separately.

Theorem 6. *There is a polynomial time algorithm with approximation ratio*

$$O(n \log\log n/(\log n)^{1/4})$$

for Steiner minimization of definite Horn formulas, where n is the number of variables in the original formula.

The algorithm uses several procedures. It uses previous algorithms for listing prime implicates of Horn formulas and for body minimization. It also uses a procedure for the exact minimization of Horn formulas having a short equivalent formula and the bipartite graph partition algorithm of Section 3.

The *prime implicate listing problem* for Horn formulas is to produce a list of all prime implicates of a Horn formula. As the number of prime implicates can be exponential in the size of the original formula, a possible criterion of efficiency is *total polynomial time*, *i.e.*, time polynomial in the combined size of the input and the output. Boros, Crama and Hammer [9] give an algorithm which lists all prime implicates of a Horn formula, in time polynomial in the size of the formula and the number of prime implicates.

Consider the following special case of (standard) Horn minimization.

Problem 1 ($\sqrt{\log n}$-Horn minimization). Given a Horn formula φ over n variables, find an equivalent minimal size Horn formula of size at most $\sqrt{\log n}$ if such a formula exists, or output *'none'*.

Lemma 7. *The $\sqrt{\log n}$-Horn minimization problem is polynomial-time solvable.*

A further ingredient of the algorithm is an efficient procedure for body minimization. The *body minimization* problem for Horn formulas asks for an equivalent Horn formula with the minimal number of distinct bodies. While Horn minimization is hard, there *are* efficient algorithms for body minimization. Such

algorithms were found in several different contexts, such as implicational systems [19] (see also [11]), functional dependencies for databases [32], directed hypergraphs [4] and computational learning theory [3].

Given a Horn formula χ over a set of variables X, we now describe a construction of a Steiner extension $\psi = \text{STEINER}(\chi)$ of χ. Let $Bodies(\chi)$ denote the set of bodies in χ, and $Heads(\chi)$ denote the set of heads in χ. Form a bipartite graph $G(\chi)$ with parts $Bodies(\chi)$ and $Heads(\chi)$, adding an edge between a body and a head if the corresponding Horn clause occurs in χ. Let $G_1, \ldots, G_t$ be a decomposition of $G(\chi)$ into bicliques obtained by the graph partition procedure of Theorem 1[4]. Let the bipartite graphs in the decomposition have parts $A_i \subseteq Bodies(\chi)$ and $B_i \subseteq Heads(\chi)$ for $i = 1, \ldots, t$. Introduce new variables $y_1, \ldots, y_t$, and let $\psi = STEINER(\chi)$ consist of the clauses $b \to y_i$ and $y_i \to h$ for every $b \in A_i$ and $h \in B_i$, $i = 1, \ldots, t$.

In the following description of the algorithm let $\sqrt{\log n}$–HORN–MIN denote the procedure of Lemma 7 and let MIN-BODY be an efficient body minimization procedure.

Input: a definite Horn formula φ

Algorithm:

if $\psi = \sqrt{\log n}$–HORN–MIN$(\varphi) \neq$ 'none' **then** return ψ
else return STEINER(MIN-BODY(φ))

The performance bound of the algorithm follows by considering different cases depending on the value of $OPT(\varphi)$ and the relationship between the number of bodies and heads returned by the body minimization procedure.

6 Restricted Horn Minimization

A special case of the Horn minimization problem is when only clauses from the original formula may be used in the new formula. Finding an irredundant subset of clauses representing the input function can always be done in polynomial time using the standard Horn formula procedures. However, there may be many irredundant formulas, having different sizes. The inapproximability result in Theorem 7 below shows that in fact it is hard to approximate the shortest one, *even if we assume that the formula to be minimized is* 3*-Horn.*

Theorem 7. *For any fixed* $0 < \epsilon < 1$*, unless* $NP \subseteq DTIME(n^{polylog(n)})$*, the restricted Horn minimization problem is* $2^{\log^{1-\epsilon} n}$*-inapproximable, even for definite* 3*-Horn formulas.*

As noted in the introduction, in the case of restricted Horn minimization one can also try to *maximize* the number of deleted clauses. We refer to this problem below as *Horn maximization.* In contrast to Theorem 7, for definite 2-Horn formulas constant approximation is possible.

[4] The bipartite graphs need not be balanced. Also, for this application it would be sufficient to consider coverings instead of partitions. The result of Section 3 is formulated for balanced partitions in order to give a stronger positive result.

Theorem 8
(a) *Both the Horn minimization and Horn maximization problems are* MAX-SNP-hard *for definite* 2*-Horn formulas without unit clauses.*
(b) *Restricted Horn minimization for definite* 2*-Horn formulas admits a* 1.5*-approximation.*
(c) *Restricted Horn maximization for definite* 2*-Horn formulas admits a* 2*-approximation.*

In view of Remark 2, these results follow from [5,42] and the correspondence between Horn formulas with binary clauses and directed graphs.

Remark 3. For **(a)**, the best inapproximability constants can be be obtained by using a randomized construction of a special class of Boolean satisfiability instances by Berman *et al.*[6] giving an inapproximability constant of $1+(1/896)$ for the minimization version and $1+(1/539)$ for the maximization version.

References

1. Albert, R., DasGupta, B., Dondi, R., Sontag, E.: Inferring (biological) signal transduction networks via transitive reductions of directed graphs. Algorithmica 51(2), 129–159 (2008)
2. Alon, N., Duke, R.A., Lefmann, H., Rödl, V., Yuster, R.: The algorithmic aspects of the regularity lemma. J. of Algorithms 16, 80–109 (1994)
3. Angluin, D., Frazier, M., Pitt, L.: Learning conjunctions of Horn clauses. Machine Learning 9, 147–164 (1992)
4. Ausiello, G., D'atri, A., Saccà, D.: Minimal representations of directed hypergraphs. SIAM J. Comput. 15(2), 418–431 (1986)
5. Berman, P., DasGupta, B., Karpinski, M.: Approximating transitive reduction problems for directed networks. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tòth, C.D. (eds.) 11th Algorithms and Data Structures Symposium. LNCS, vol. 5664, pp. 74–85. Springer, Heidelberg (2009)
6. Berman, P., Karpinski, M., Scott, A.D.: Approximation hardness of short symmetric instances of MAX-3SAT. Electronic Colloquium on Computational Complexity, Report TR03-049 (2003)
7. Boros, E.: Horn functions. In: Crama, Y., Hammer, P.L. (eds.) Boolean Functions: Theory, Algorithms and Applications, Cambridge Univ. Press, Cambridge (forthcoming, 2010)
8. Boros, E., Čepek, O.: On the complexity of Horn minimization, Rutcor Research Report 1-94 (1994)
9. Boros, E., Crama, Y., Hammer, P.L.: Polynomial-time inference of all valid implications for Horn and related formulae. Ann. Math. Artif. Intell. 1, 21–32 (1990)
10. Bublitz, S.: Decomposition of graphs and monotone formula size of homogeneous fuctions. Acta Informatica 23, 689–696 (1996)
11. Caspard, N., Monjardet, B.: The lattice of closure systems, closure operators and implications on a finite set: a survey. Discrete Appl. Math. 127, 241–269 (2003)
12. Chandra, A.K., Markowsky, G.: On the number of prime implicants. Discrete Math. 24, 7–11 (1978)

13. Chung, F.R.K., Erdős, P., Spencer, J.: On the decomposition of graphs into complete bipartite graphs. Studies in Pure Mathematics, To the Memory of Paul Turán, pp. 95-101. Akadémiai Kiadó (1983)
14. Clote, P., Kranakis, E.: Boolean Functions and Models of Computation. Springer, Heidelberg (2003)
15. Cook, S.A.: Feasibly constructive proofs and the propositional calculus. In: STOC, pp. 83–97 (1975)
16. Feige, U., Kogan, S.: Hardness of approximation of the balanced complete bipartite subgraph problem, Tech. Rep. MCS04-04, Dept. of Comp. Sci. and Appl. Math., The Weizmann Inst. of Science (2004)
17. Flögel, A., Kleine Büning, H., Lettmann, T.: On the restricted equivalence subclasses of propositional logic. ITA 27, 327–340 (1993)
18. Frazier, M.D.: Matters Horn and other features in the computational learning theory landscape: the notion of membership, Ph.D. thesis, Univ. of Illinois at Urbana-Champaign (1994)
19. Guigues, J.L., Duquenne, V.: Familles minimales d'implications informatives résultant d'un tableau de données binaires. Math. Sci. Humaines 95, 5–18 (1986)
20. Hammer, P.L., Kogan, A.: Horn functions and their DNFs. Information Processing Letters 44, 23–29 (1992)
21. Hammer, P.L., Kogan, A.: Optimal compression of propositional Horn knowledge bases: complexity and approximation. Artificial Intelligence 64, 131–145 (1993)
22. Hammer, P.L., Kogan, A.: Quasi-acyclic Horn knowledge bases: optimal compression. IEEE Trans. on Knowledge and Data Engineering 7, 751–762 (1995)
23. Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem. North-Holland, Amsterdam (1992)
24. Kirchner, S.: Lower bounds for Steiner tree algorithms and the construction of bicliques in dense graphs, Ph.D. Dissertation, Humboldt-Universität zu Berlin (2008) (in German)
25. Kleine Büning, H., Lettmann, T.: Propositional Logic: Deduction and Algorithms. Cambridge Univ. Press, Cambridge (1999)
26. Kortsarz, G., Krauthgamer, R., Lee, J.R.: Hardness of approximating vertex-connectivity network design problems. SIAM J. Comput. 33(3), 704–720 (2004)
27. Kővári, T., Sós, V.T., Turán, P.: On a problem of K. Zarankiewicz. Colloq. Math. 3, 50–57 (1954)
28. Langlois, M., Sloan, R.H., Turán, G.: Horn upper bounds and renaming. J. Satisfiability, Boolean Modelling and Computation 7, 1–15 (2009)
29. Langlois, M., Sloan, R.H., Szörényi, B., Turán, G.: Horn complements: towards Horn-to-Horn belief revision, pp. 466–471. AAAI, Menlo Park (2008)
30. Langlois, M., Mubayi, D., Sloan, R.H., Turán, G.: Combinatorial Problems for Horn Clauses. In: Lipshteyn, M., Levit, V.E., McConnell, R.M. (eds.) Graph Theory, Computational Intelligence and Thought. LNCS, vol. 5420, pp. 54–65. Springer, Heidelberg (2009)
31. Lovász, L.: Coverings and colorings of hypergraphs. In: Proc. 4th Southeastern Conf. on Combinatorics, Graph Theory and Computing, Utilitas Math., pp. 3–12 (1973)
32. Maier, D.: The Theory of Relational Databases. Comp. Sci. Press, Rockville (1983)
33. Mubayi, D., Turán, G.: Finding bipartite subgraphs efficiently. Inf. Proc. Lett. 110, 174–177 (2010)
34. Orlin, J.: Contentment in graph theory: covering graphs with cliques. Indag. Math. 80, 406–424 (1977)

35. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall, Englewood Cliffs (2002)
36. Singh, P., Lin, T., Mueller, E.T., Lim, G., Perkins, T., Zhu, W.L.: Open Mind Common Sense: knowledge acquisition from the general public., CoopIS/DOA/ODBASE, 1223–1237 (2002)
37. Tarján, T.: Complexity of lattice-configurations. Studia Sci. Math. Hung. 10, 203–211 (1975)
38. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Seminars in Mathematics, V. A. Steklov Mathematical Institut, Leningrad, vol. 8 (1968)
39. Tuza, Z.: Covering of graphs by complete bipartite subgraphs; complexity of 0-1 matrices. Combinatorica 4, 111–116 (1984)
40. Umans, C.: Hardness of approximating Σ_2^p minimization problems. In: FOCS, pp. 465-474 (1999)
41. Umans, C.: The Minimum Equivalent DNF problem and shortest implicants. J. Comput. Syst. Sci. 63, 597–611 (2001)
42. Vetta, A.: Approximating the minimum strongly connected subgraph via a matching lower bound. In: 12th ACM-SIAM Symposium on Discrete Algorithms, pp. 417–426 (2001)

Choosing, Agreeing, and Eliminating in Communication Complexity*

Amos Beimel[1], Sebastian Ben Daniel[1], Eyal Kushilevitz[2], and Enav Weinreb[2]

[1] Dept. of Computer Science, Ben Gurion University, Beer Sheva, Israel
[2] Dept. of Computer Science, Technion, Haifa, Israel

Abstract. We consider several questions inspired by the direct-sum problem in (two-party) communication complexity. In all questions, there are k fixed Boolean functions $f_1, \ldots, f_k$ and Alice and Bob have k inputs $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$, respectively. In the *eliminate* problem, Alice and Bob should output a vector $\sigma_1, \ldots, \sigma_k$ such that $f_i(x_i) \neq \sigma_i$ for at least one i (i.e., their goal is to eliminate one of the 2^k output vectors); in *choose*, Alice and Bob should return $(i, f_i(x_i, y_i))$ and in *agree* they should return $f_i(x_i, y_i)$, for some i. The question, in each of the three cases, is whether one can do better than solving one (say, the first) instance. We study these three problems and prove various positive and negative results.

1 Introduction

A basic question in complexity theory is how the complexity of computing k independent instances relates to the complexity of computing one instance. Such problems, called *direct sum problems*, have been studied for a variety of computational models. Broadly, the direct sum question asks (with respect to an arbitrary computational model and any complexity measure):

Question 1. Can "solving" k functions $f_1, \ldots, f_k$ on k independent inputs $x_1, \ldots, x_k$ (respectively) be done more "efficiently" than just "solving" each $f_i(x_i)$?

(Of particular interest is the special case where all functions are identical.) Since the inputs are independent, it is tempting to conjecture that in reasonable models the answer is negative. Indeed, it was proved that in several models no significant saving can be obtained; e.g., for decision trees [7,22,27,14]. However, for other models, some savings are possible despite the independence of the inputs, e.g., non-deterministic communication complexity and randomized communication complexity [16,10], deterministic communication complexity of relations [10], and distributional communication complexity [27]. For other models, the answer is still unknown; e.g., in circuit complexity [11,23,29]. Direct sum results are important for understanding the power of a computational model. For example,

* The first author is supported by ISF grant 938/09. The second author is partially supported by the Frankel Center for Computer Science. The third and fourth authors are supported by ISF grant 1310/06.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 451–462, 2010.

it was shown in [17] that a negative answer for a variant of this question implies circuit lower bounds, in particular, $\mathcal{NC}^1 \neq \mathcal{NC}^2$. Furthermore, direct sum results on the information complexity have been used to prove lower bounds on the communication complexity of functions [4].

To better understand direct sum questions, a simpler task has been proposed – *eliminating* a vector of answers. More precisely, for k fixed Boolean functions $f_1, \ldots, f_k$, given k inputs $x_1, \ldots, x_k$, find a vector $\sigma_1, \ldots, \sigma_k$ such that $f_i(x_i) \neq \sigma_i$ for at least one i. In other words, given $x_1, \ldots, x_k$, a-priori there are 2^k possible vectors of outputs for the k instances. Solving the direct sum problem is finding the correct vector of outputs; eliminating means returning one of the $2^k - 1$ vectors which is not the vector of outputs. Clearly, if we solve one instance and obtain, say, $a_1 = f_1(x_1)$, then we can eliminate the vector $\overline{a}_1, \sigma_2, \ldots, \sigma_k$ for any $\sigma_2, \ldots, \sigma_k$. The question is if we can do better; that is,

Question 2. Can solving `eliminate`, on k independent instances $x_1, ..., x_k$ of k functions $f_1, \ldots, f_k$, be "easier" than solving the "easiest" function?

(Again, of a particular interest is when all functions are identical.) This question and related ones were studied in the context of polynomial-time computation [1,3,8,9,13,19,24,25,26,28] and computation in general [5,6,12,15,20]. The question was explored for communication complexity by Ambainis et al. [2].

In this work, we introduce two new problems related to the direct sum question, `choose` and `agree`. As before, there are k fixed Boolean functions $f_1, \ldots, f_k$, and we are given k instances $x_1, \ldots, x_k$. In the `choose` problem, the task is to solve one instance of our choice; i.e., return $(i, f_i(x_i))$, for some i. Intuitively, `choose` is allowed to pick an "easy" input and answer it. The question is:

Question 3. Can solving `choose`, on k independent instances $x_1, ..., x_k$ of k functions $f_1, \ldots, f_k$, be "easier" than solving the "easiest" function?

In the `agree` problem, the task is to return $f_i(x_i)$ for some i (possibly without knowing for which instance i it corresponds). That is, if all outputs agree, i.e., $f_1(x_1) = f_2(x_2) = \cdots = f_k(x_k) = \sigma$, then $\texttt{agree}(x_1, \ldots, x_k)$ must return σ, otherwise it may return either 0 or 1. The question is:

Question 4. Can solving `agree`, on k independent instances $x_1, ..., x_k$ of k functions $f_1, \ldots, f_k$, be "easier" than solving the "easiest" function?

Comparing the three tasks, `choose` is the hardest and `eliminate` is the easiest, as solving `choose` implies solving `agree` which, in turn, implies solving `eliminate` (if we get an answer σ for $\texttt{agree}(x_1, \ldots, x_n)$, then we can eliminate the output vector $(\overline{\sigma}, \ldots, \overline{\sigma})$). However, eliminating may potentially be much easier than `choose` and `agree`; for example, if $f_1 = f_2$ and $x_1 = x_2$, then solving `agree` implies solving $f_1(x_1)$, while for `eliminate` one may return $(0, 1)$ (or $(1, 0)$) without any computation. Furthermore, if we can solve `choose` efficiently, then we can solve the direct sum efficiently (use `choose` to solve one instance and solve the other instances independently). We do not know of any connections between the direct sum problem and `agree` or `eliminate`.

We start by mentioning some related work. The direct sum question in communication complexity for deterministic protocols of functions is still open. There is an example of a relation in which saving of $O(k \cdot \log n)$ bits is possible for k instances by a deterministic protocol [10]. For non-deterministic protocols and randomized protocols, some saving is possible for some functions [10,16]. However, the best possible "generic" upper bound for the direct sum of randomized protocols is not known (while in the non-deterministic case, an additive $O(\log n)$ savings is the best possible [10,16]). Ambainis et al. [2] study the communication complexity of `eliminate`. They conjecture that, for functions, no saving is possible for deterministic protocols. To support their conjecture they supply, in addition to other results, lower bounds for deterministic, non-deterministic, and randomized protocols for `eliminate` of specific functions.

Our Results. We define the `choose` and `agree` problems and study their properties, as well as the properties of `eliminate`.

- For randomized public-coin protocols, we show that saving of $O(\log k)$ bits is possible for `eliminate` of some functions (e.g., the inner-product function).
- On the negative side, we prove that the randomized communication complexity of solving a function f is a lower bound on the randomized communication complexity of $\texttt{eliminate}_{f^k}$, i.e., on computing eliminate on k instances of f. This implies, together with a trivial upper bound, that the randomized communication complexity of computing a function f characterizes the randomized communication complexity of computing $\texttt{eliminate}_{f^k}$, up-to a factor of $2^{O(k)}$. In particular, our results show that eliminate of IP requires $n - O(k)$ bits even for randomized protocols. This improves a lower bound of $n/(2^{O(k)} \log n \log\log n)$ for randomized protocols for IP proved in [2].
- We relate the complexity of $\texttt{choose}_{f,g}$ to the non-deterministic and deterministic complexity of solving f and g. In particular, we show that if the non-deterministic communication complexity of f and g are high, then the deterministic communication complexity of $\texttt{choose}_{f,g}$ is high. This implies that for most functions the communication complexity of $\texttt{choose}_{f,g}$ is $\Omega(n)$. We prove a similar, however weaker, lower bound for $\texttt{agree}_{f,g}$.

To better understand if saving is possible for `eliminate` in communication complexity, we explore a restriction of eliminate, called `r-eliminate`. In this case, Alice has k inputs $x_1, \ldots, x_k$, however, Bob has *one* input y. The goal of Alice and Bob is to find a vector $\sigma_1, \ldots, \sigma_k$ such that $f_i(x_i, y) \neq \sigma_i$, for at least one i. In the rest of this section, we assume that $f_1 = f_2 = \ldots = f_k = f$. In this model significant saving is possible even for $k = 2$. For example for `r-eliminate` of the equality function, if Alice holds two inputs $x_1 = x_2$, she can eliminate, say, $(0, 1)$ without any communication, and if $x_1 \neq x_2$, she can eliminate $(1, 1)$ without any communication. We show that for some other functions `r-eliminate` is hard and we also give additional examples where some saving is possible:

- `r-eliminate` of the disjointness function on k instances can be computed using a deterministic protocol sending $(n \log k)/k$ bits; that is, better than

the deterministic complexity of solving one instance of disjointness (which is n). By [2], `eliminate` of k instances of disjointness requires $n - O(\log n)$ bits deterministically [2]. Using this result, we prove that `r-eliminate` of disjointness on k instances requires $\Omega(n/k)$ bits deterministically. That is, our protocol is optimal up to a factor of $\log k$.
- `r-eliminate` of the inner-product function on k instances can be solved deterministically by sending $n - k + 2$ bits. Thus, some saving is possible for large k's. We show that our lower bound for $\texttt{eliminate}_{\mathrm{IP}^k}$ can be translated to a lower bound of $\Omega(n/k)$ for `r-eliminate` on k instances of IP for randomized protocols.
- For most functions, `r-eliminate` of two instances requires at least $n - 5$ bits. Thus, the naive protocol where Bob sends his input to Alice is nearly optimal.

In the full version of this paper, we also consider decision trees and circuit complexity. We show that for decision trees, no saving can be obtained for `choose`, `agree`, and `eliminate`. That is, any decision tree solving $\texttt{eliminate}_{f,g}$ can be converted into a decision tree of the same size and depth that either computes f or computes g. This generalizes the results that no saving can be achieved for decision trees in the direct sum problem [7,22,27,14]. We also prove that the size of the smallest circuit solving $\texttt{agree}_{f,g}$ is equal to the deterministic communication complexity of $\texttt{choose}_{R_f,R_g}$, where R_f denotes the Karchmer-Wigderson relation related to f [18]. This is a generalization of results of [18] on the relation between circuit size of a function f and the communication complexity of R_f.

2 Preliminaries

We consider the two-party communication complexity model of Yao [30]. In this model, there are three finite sets X, Y, and Z, and a relation $R \subseteq X \times Y \times Z$. Two players, Alice and Bob, get $x \in X$ and $y \in Y$ respectively. Their goal is to compute z such that $(x, y, z) \in R$ by exchanging bits according to some protocol (we assume that for every x, y there is some z such that $(x, y, z) \in R$). Let $\mathcal{D}(R)$ be the deterministic communication complexity of solving R, $\mathcal{N}(R)$ be the nondeterministic communication complexity, $\mathcal{N}^0(R)$ and $\mathcal{N}^1(R)$ be the one-sided nondeterministic communication complexities, and $\mathcal{R}_\epsilon(R)$ and $\mathcal{R}^{\mathrm{pub}}_\epsilon(R)$ be its ϵ-error randomized communication complexity with private and public random string, respectively. For formal definitions, see [21].

Next, we define the three problems inspired by the direct sum question. For the three problems there are several possible outputs, i.e., they are relations.

Definition 1 (Choose, Agree, and Eliminate). *Let* $f_1, \ldots, f_k : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *be functions. In all three problems, the input of Alice and Bob are a* k*-tuple* $x_1, \ldots, x_k$ *and* $y_1, \ldots, y_k$ *respectively. In* $\texttt{choose}_{f_1,\ldots,f_k}$ *an output is* $(i, f_i(x_i, y_i))$ *where* $i \in \{1, \ldots, k\}$*. In* $\texttt{agree}_{f_1,\ldots,f_k}$*, an output is* $f_i(x_i, y_i)$ *where* $i \in \{1, \ldots, k\}$*. Finally, in* $\texttt{eliminate}_{f_1,\ldots,f_k}$*, an output is any vector* $\sigma_1, \ldots, \sigma_k \in \{0,1\}$ *for which there exists* i *such that* $\sigma_i \neq f_i(x_i, y_i)$*.*

If $f_1 = ... = f_k = f$ we abbreviate $f_1, \ldots, f_k$ by f^k. Note that $\mathcal{R}_\epsilon(\texttt{eliminate}_{f^k}) = O(1)$ for $\epsilon \geq 1/2^k$ as a random k-bit output will err with probability $1/2^k$. As we shall see that, for $\epsilon < 1/2^k$, $\mathcal{R}_\epsilon(\texttt{eliminate}_{f^k})$ is large for some functions.

Next, we define a restricted version of `eliminate` where Bob gets the same y for all k functions. For simplicity, assume that all k functions are equal.

Definition 2 (R-Eliminate). *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function. In $\texttt{r-eliminate}_{f^k}$, Alice gets a k-tuple $x_1, \ldots, x_k$ and Bob gets a* single *input y; an output is any vector $\sigma_1, \ldots, \sigma_k \in \{0,1\}$ for which there exists i such that $\sigma_i \neq f(x_i, y)$.*

The next theorem relates the randomized communication complexity and the distributional communication complexity of f.

Theorem 1 (Yao's min-max Theorem [30]). *Let $f : X \times Y \to \{0,1\}$. Then, $\mathcal{R}^{\text{pub}}_\epsilon(f) = \max_\mu \mathcal{D}^\mu_\epsilon(f)$, where $\mathcal{D}^\mu_\epsilon$ denotes the ϵ-error* distributional *complexity with respect to μ and the maximum is taken over all distributions μ on $X \times Y$.*

3 General Bounds for Choose, Agree, and Eliminate

In this section, we show that saving is possible in public-coin randomized protocols for eliminate of some functions. We then prove lower bounds on the complexity of $\texttt{choose}_{f_1,f_2}$, $\texttt{agree}_{f_1,f_2}$, and $\texttt{eliminate}_{f^k}$.

Theorem 2. *There exist a randomized protocol for $\texttt{eliminate}_{f^k}$ with complexity $\max\{n - 0.5\log k, k\log k\} + O(1)$ for $\epsilon = 2^{-k}/e$, in which at least one of the parties knows the answer at the end.*

Proof. We describe a protocol with the desired complexity. In the first step of the protocol, Bob checks if he has at least $\sqrt{k}$ distinct inputs (among his k inputs). If so, both Alice and Bob treat the public random string r as a sequence of blocks $r_1, r_2, \ldots$, each of length n. If there exists i and j such that $y_i = r_j$, among the first $2^n/\sqrt{k}$ blocks of r, Bob sends j to Alice. In this case, Alice computes $\sigma_\ell = 1 - f(x_\ell, r_j)$ for $1 \leq \ell \leq k$ and outputs $\sigma_1, \ldots, \sigma_k$. If Bob cannot find such index, he sends 0 to Alice, who outputs a random k-bit string. If Bob has less than $\sqrt{k}$ distinct inputs, and Alice has at least $\sqrt{k}$ distinct inputs, they reverse roles. In this case, Bob gets the output. In both cases, the communication complexity is $O(1) + \log(2^n/\sqrt{k}) = n - 0.5\log k + O(1)$.

If both Alice and Bob have less than $\sqrt{k}$ distinct input values (they discover this fact using O(1) communication) then they do the following. Since Bob has less than $\sqrt{k}$ values, there is a value that appears more than $\sqrt{k}$ times. Bob sends to Alice $\sqrt{k}$ indices in which his inputs are the same. Since Alice has less than $\sqrt{k}$ values, there are two indices i, j among the indices that Bob sent such that $x_i = x_j$. Alice outputs an arbitrary vector $(\sigma_1, \ldots, \sigma_k)$ such that $\sigma_i \neq \sigma_j$ which is always correct, and the communication complexity is $\sqrt{k}\log k + O(1)$.

To complete the analysis of the protocol, we bound the error probability for the cases that Bob (or Alice) has at least $\sqrt{k}$ distinct input values. The

probability that $y_i \neq r_j$ for all $1 \leq i \leq k$ and $1 \leq j \leq 2^n/\sqrt{k}$ is less than $(1-1/2^n)^{\sqrt{k}2^n/\sqrt{k}} \leq 1/e$. In this case, the protocol errs with probability $1/2^k$. If $y_i = r_j$ then the protocol never errs. Thus, the error probability of the protocol is $2^{-k}/e$. □

Next we state lower bounds for $\mathtt{choose}_{f_1,f_2}$ and $\mathtt{agree}\, f_1, f_2$ in terms of the deterministic and non-deterministic communication complexity of f_1 and f_2, the proof for choose appears in the final version.

Theorem 3. $\mathcal{D}(\mathtt{choose}_{f_1,f_2}) \geq \min\{\mathcal{D}(f_1), \mathcal{D}(f_2), \max\{\mathcal{N}(f_1), \mathcal{N}(f_2)\}\}$ *for every two functions* f_1, f_2.

Theorem 4. *For every two functions* $f_1, f_2 : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, $\mathcal{D}(\mathtt{agree}_{f_1,f_2}) \geq \max\left\{\min\left\{\mathcal{N}^1(f_1), \mathcal{N}^0(f_2)\right\}, \min\left\{\mathcal{N}^0(f_1), \mathcal{N}^1(f_2)\right\}\right\} - \log(2n)$.

Proof. Let $\mathcal{P}$ be a (deterministic) protocol for $\mathtt{agree}_{f_1,f_2}$ whose complexity is $\mathcal{D}(\mathtt{agree}_{f_1,f_2})$. We construct a non-deterministic protocol with communication complexity $\mathcal{D}(\mathtt{agree}_{f_1,f_2}) + \log(2n)$ either for proving $f_1(x,y) = 0$ or for proving $f_2(x,y) = 1$. Let $S_1 \subseteq f_1^{-1}(0)$ and $S_2 \subseteq f_2^{-1}(1)$. A pair of inputs $(x_2, y_2) \in S_2$ is an f_1-loser if $\mathcal{P}(x_1, x_2, y_1, y_2) = 0$ for at least $|S_1|/2$ pairs $(x_1, y_1) \in S_1$.

Proposition 1. *For every two sets* $S_1 \subseteq f_1^{-1}(0)$ *and* $S_2 \subseteq f_2^{-1}(1)$*, either there is a pair* $(x_2, y_2) \in S_2$ *that is an* f_1*-loser or a pair* $(x_1, y_1) \in S_1$ *that is an* f_2*-loser.*

Proposition 2. *Either there is a sequence of* $2n$ *pairs* $(x^1, y^1), \ldots, (x^{2n}, y^{2n}) \in f_2^{-1}(1)$ *s.t., for every* $(x,y) \in f_1^{-1}(0)$*, it holds that* $\mathcal{P}(x, x^i, y, y^i) = 0$ *for at least one* i*, or there is a sequence of* $2n$ *pairs* $(x^1, y^1), \ldots, (x^{2n}, y^{2n}) \in f_1^{-1}(0)$ *s.t., for every* $(x,y) \in f_2^{-1}(1)$*, it holds that* $\mathcal{P}(x^i, x, y^i, y) = 1$ *for at least one* i*.*

Assume that the first case of Proposition 2 holds. We construct a non-deterministic protocol for proving that $f_1(x,y) = 0$ (if the second case holds, we would construct a non-deterministic protocol for proving $f_2(x,y) = 1$). Let $(x^1, y^1), \ldots, (x^{2n}, y^{2n})$ be the sequence guaranteed by the proposition. The first idea is, given inputs x, y to f_1, to execute $\mathcal{P}(x, x^i, y, y^i)$ for $i = 1, \ldots, 2n$. If in at least one of the executions, the output of $\mathcal{P}$ is 0 then clearly $f_1(x,y) = 0$. Furthermore, if $f_1(x,y) = 0$, then at least one of the executions will return 0. This protocol activates $\mathcal{P}$ $2n$ times, and is, thus, extremely inefficient. However, Alice can guess an index i such that $\mathcal{P}(x, x^i, y, y^i) = 0$, send it to Bob, and Alice and Bob execute $\mathcal{P}(x, x^i, y, y^i)$, and output 0 iff $\mathcal{P}$ outputs 0. Therefore, $\mathcal{D}(\mathtt{agree}_{f_1,f_2}) + \log 2n \geq \min\left\{\mathcal{N}^0(f_1), \mathcal{N}^1(f_2)\right\}$. Similarly, $\mathcal{D}(\mathtt{agree}_{f_1,f_2}) + \log 2n \geq \min\left\{\mathcal{N}^1(f_1), \mathcal{N}^0(f_2)\right\}$. □

Theorem 4 does not rule out an exponential gap between $\mathcal{D}(\mathtt{agree}_{f_1,f_2})$ and $\min\{\mathcal{D}(f_1), \mathcal{D}(f_2)\}$. For the special case, $\mathtt{agree}_{f,\overline{f}}$, Theorem 4 implies that the gap is at most a quadratic; i.e., $\mathcal{D}(\mathtt{agree}_{f,\overline{f}}) \geq \mathcal{N}(f) - \log(2n) - 1 \geq \Omega(\sqrt{\mathcal{D}(f)}) - \log(2n)$. This should be compared, on one hand, to $\mathtt{choose}_{f,\overline{f}}$, which is as hard as computing f and, on the other hand, to solving $\mathtt{eliminate}_{f,\overline{f}}$, which is equal to solving $\mathtt{eliminate}_{f^2}$. Furthermore, $\mathtt{agree}_{f,f}$ is equivalent to computing f.

We end this section by stating a lower bound for $\texttt{eliminate}_{f^k}$ using the randomized communication complexity of f.

Theorem 5. *$\mathcal{R}^{\text{pub}}_{\epsilon}(\texttt{eliminate}_{f^k}) \geq \mathcal{R}^{\text{pub}}_{\epsilon'}(f)$, where $\epsilon' = \frac{1}{2} - \frac{1/2 - \epsilon 2^{k-1}}{2^k - 1}$.*

Proof. We prove the lower bound using Yao's min-max Theorem (Theorem 1). Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function and μ be a distribution on $\{0,1\}^n \times \{0,1\}^n$ such that $\mathcal{R}^{\text{pub}}_{\epsilon}(f) = \mathcal{D}^{\mu}_{\epsilon}(f)$. We start with a randomized protocol $\mathcal{P}$ for $\texttt{eliminate}_{f^k}$ with complexity $\mathcal{R}_{\epsilon}(\texttt{eliminate}_{f^k})$. We construct a deterministic protocol $\mathcal{P}''$ with the same complexity as $\mathcal{P}$ such that the probability over the inputs, according to μ, that $\mathcal{P}''$ computes $f(x,y)$ correctly is at least ϵ'. Thus, $\mathcal{R}_{\epsilon}(\texttt{eliminate}_{f^k}) \geq \mathcal{D}^{\mu}_{\epsilon'}(f) = \mathcal{R}^{\text{pub}}_{\epsilon'}(f)$. The construction of $\mathcal{P}''$ is done in stages. We first define a protocol $\mathcal{P}'$ and use it to define $\mathcal{P}''$.

For the construction of $\mathcal{P}''$, we use constants $q_0, \ldots, q_k$ such that $1 = q_0 \geq q_1 \geq \cdots \geq q_k = \epsilon$. We also use random variables $Z_1, \ldots, Z_k$ defined as follows. Pick inputs $x_1, y_1, \ldots, x_k, y_k$ according to the distribution μ^k, execute $\mathcal{P}(x_1, \ldots, x_k, y_1, \ldots, y_k)$ and let $\sigma_1, \ldots, \sigma_k$ be its output. Finally, for $1 \leq i \leq k$, define $Z_i = T$ if $\sigma_i = f(x_i, y_i)$ and $Z_i = F$ otherwise.

By the correctness of $\mathcal{P}$, $\Pr[Z_1 = \cdots = Z_k = T] \leq \epsilon = q_k$, where the probability is taken over the choice of inputs, according to the distribution μ^k, and over the randomness of $\mathcal{P}$. Thus, there must be an index i, where $1 \leq i \leq k$, such that $\Pr[Z_i = T | Z_1 = \cdots = Z_{i-1} = T] \leq q_i/q_{i-1}$. Let i be the smallest such index. That is, $\Pr[Z_j = T | Z_1 = \cdots = Z_{j-1} = T] \geq q_j/q_{j-1}$ for $1 \leq j \leq i-1$. In particular, $p \stackrel{\text{def}}{=} \Pr[Z_1 = \cdots = Z_{i-1} = T] = \prod_{1 \leq j \leq i-1} \Pr[Z_j = T | Z_1 = \cdots = Z_{j-1} = T] \geq (q_1/q_0) \cdot (q_2/q_1) \cdot \ldots \cdot (q_{i-1}/q_{i-2}) = q_{i-1}$.

Using this index i, we construct a randomized protocol $\mathcal{P}'$ for computing f. Intuitively, $\mathcal{P}'$ will use the fact that with a noticeable probability it can take the i-th output of $\mathcal{P}$ and invert it and obtain a correct output for the i-th pair of inputs (assuming they are distributed according to μ). On input, x, y, the protocol $\mathcal{P}'$ samples $x_1, y_1, \ldots, x_{i-1}, y_{i-1}, x_{i+1}, y_{i+1}, \ldots, x_k, y_k$ according to μ^{k-1} and gives these inputs both to Alice and to Bob (we will see later how to implement this step without communication). Alice and Bob execute $\mathcal{P}(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x} = (x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k)$ and $\boldsymbol{y} = (y_1, \ldots, y_{i-1}, y, y_{i+1}, \ldots, y_k)$; let $\sigma_1, \ldots, \sigma_k$ be the output of $\mathcal{P}$. If $\sigma_j = f(x_j, y_j)$ for $1 \leq j \leq i-1$, then Alice and Bob output $\overline{\sigma}_i$. Otherwise, Alice chooses a random bit b with uniform distribution and outputs this bit.

We next claim that if the inputs x, y are distributed according to μ, then the error of $\mathcal{P}'$ is at most ϵ'. Protocol $\mathcal{P}'$ succeeds in two cases: (1) $\sigma_j = f(x_j, y_j)$ for $1 \leq j \leq i-1$ and $\sigma_i \neq f(x,y)$; by our choice of i, this happens with probability at least $p \cdot (1 - q_i/q_{i-1})$ and (2) $\sigma_j \neq f(x_j, y_j)$ for some $1 \leq j \leq i-1$ and $b = f(x,y)$; this happens with probability $0.5(1-p)$. All together the success probability of $\mathcal{P}'$ is $p \cdot (1 - q_i/q_{i-1}) + 0.5(1-p)$, where the probability is taken over the choice of the inputs x, y according to μ, the choice of the other $k-1$ pairs of inputs for $\mathcal{P}$ according to μ^{k-1}, the randomness of $\mathcal{P}$, and the choice of b. Since $p \leq q_{i-1}$ and by choosing q_i, q_{i-1} such that $q_i/q_{i-1} \leq 0.5$, the success probability is at least $q_{i-1} \cdot (1 - q_i/q_{i-1}) + 0.5(1 - q_{i-1}) = 0.5 + 0.5 q_{i-1} - q_i$. We

choose $q_0 \stackrel{\text{def}}{=} 1$ and $q_j \stackrel{\text{def}}{=} \frac{2^{j-1}}{2^j-1}\epsilon' - (1 - \frac{1}{2^j-1})$ for $1 \leq j \leq k$. Notice that $q_1 = \epsilon'$, $q_k = \epsilon$, and the success probability of $\mathcal{P}'$ is at least $1 - \epsilon'$.

Protocol $\mathcal{P}'$ is randomized and we want a deterministic protocol $\mathcal{P}''$. Furthermore, we need to explain how we can assume that Alice and Bob know all the other $k-1$ pairs of inputs. The derandomization of $\mathcal{P}'$ is done using a simple averaging argument: there exists a random string for $\mathcal{P}'$ such that the success probability of $\mathcal{P}'$ with this random string is at least $1 - \epsilon'$, where now the success probability is taken over the choice of x, y according to μ. We fix such random string to obtain $\mathcal{P}''$. As this random string contains $x_1, y_1, \ldots, x_{i-1}, y_{i-1}, x_{i+1}, y_{i+1}, \ldots, x_k, y_k$, and $\mathcal{P}''$ executes $\mathcal{P}'$ with the fixed random string, both Alice and Bob know these inputs. Thus, as the communication complexity of $\mathcal{P}''$ is the same as the communication complexity of $\mathcal{P}$, the theorem follows. □

Together with a simple protocol for eliminate, which solves one instance and guesses the other coordinates, we have that $\mathcal{R}_{\epsilon \cdot 2^{k-1}}(f) \geq \mathcal{R}_\epsilon(\texttt{eliminate}_{f^k}) \geq \mathcal{R}_{1/2 - \frac{1/2 - \epsilon 2^{k-1}}{2^k-1}}(f) \geq 2^{-O(k)}\mathcal{R}_\epsilon(f)$ for $\epsilon < 1/2^k$. In other words, $\mathcal{R}_\epsilon(f)$ characterizes $\mathcal{R}_\epsilon(\texttt{eliminate}_{f^k})$ up to a factor of $2^{O(k)}$.

4 Eliminate and R-Eliminate

We consider the `r-eliminate` task, where Alice gets a sequence of inputs $x_1, \ldots, x_k$ and Bob gets a single input y, and their goal is to compute some impossible outcome for $f(x_1, y), \ldots, f(x_k, y)$. In this model, we have the largest gap possible between $\mathcal{D}(f)$ and $\mathcal{D}(\texttt{r-eliminate}_{f^2})$.

Example 1. Consider the equality function, where $\mathrm{EQ}(x, y) = 1$ iff $x = y$. It is known that $\mathcal{D}(\mathrm{EQ}) = n$. However, $\mathcal{D}(\texttt{r-eliminate}_{\mathrm{EQ}^2}) = O(1)$: if $x_1 = x_2$ Alice outputs $(0, 1)$ (if $x_1 = y$ then so is x_2). Otherwise, if $x_1 \neq x_2$, Alice outputs $(1, 1)$ (since x_1, x_2 cannot both equal y). By [2], $\mathcal{D}(\texttt{eliminate}_{\mathrm{EQ}^k}) = \Omega(n)$, thus we also get the largest possible separation between `r-eliminate` and `eliminate`.

Eliminate vs. R-Eliminate. We next identify a simple property of functions that enables proving that the communication complexity of `r-eliminate` and `eliminate` can differ by a factor of at most $1/k$ for these functions.

Definition 3. *A function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is padable with respect to a function $g : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$, where $m > n$, if there exists a value $b \in \{0,1\}$ such that $f(x, y) = g(b^i \circ x \circ b^{m-n-i}, a \circ y \circ a')$, for every $i \leq m - n$, every $x, y \in \{0,1\}^n$, and every $a \in \{0,1\}^i$, $a' \in \{0,1\}^{m-n-i}$.*

We shall see that natural functions, e.g., disjointness and inner-product, are padable with respect to themselves (by taking $b = 0$).

Lemma 1. *If a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is padable with respect to $g : \{0,1\}^{nk} \times \{0,1\}^{nk} \to \{0,1\}$, then $\mathcal{D}(\texttt{r-eliminate}_{g^k}) \geq \mathcal{D}(\texttt{eliminate}_{f^k})$ and $\mathcal{R}_\epsilon(\texttt{r-eliminate}_{g^k}) \geq \mathcal{R}_\epsilon(\texttt{eliminate}_{f^k})$.*

Proof. Let $x_1, \ldots, x_k \in \{0,1\}^n$ and $y_1, \ldots, y_k \in \{0,1\}^n$ be the inputs of Alice and Bob (respectively) for $\texttt{eliminate}_{f^k}$. Let $y' = y_1 \circ \cdots \circ y_k$. As f is padable with respect to g (with some b), we can pad each input x_i of Alice to get $x'_i = b^{n(i-1)} x_i b^{n(k-i)}$ (of length nk). We execute an optimal deterministic protocol for $\texttt{r-eliminate}_{g^k}$ on inputs $(x'_1, \ldots x'_k, y')$, and let τ be the answer of the eliminate protocol. Since f is padable with respect to g, we have $f(x_i, y_i) = g(x'_i, y')$ and so the answer τ is a possible answer for $\texttt{eliminate}_{f^k}(x_1, \ldots, x_k, y_1, \ldots, y_k)$ as well. Hence, $\mathcal{D}(\texttt{eliminate}_{f^k}) \leq \mathcal{D}(\texttt{r-eliminate}_{g^k})$. The same reduction applies for the randomized case. □

R-Eliminate for the Disjointness Function. Let DISJ denote the disjointness function, namely $\mathrm{DISJ}(S,T) = 1$ iff $S \cap T = \emptyset$ for $S, T \subseteq [n]$ (the inputs sets S, T are represented by their n-bit characteristic vectors).

Theorem 6. $\mathcal{D}(\texttt{r-eliminate}_{\mathrm{DISJ}^k}) = O(\frac{n}{k} \cdot \log k)$.

Proof. Let $S_1, \ldots, S_k$ be the inputs of Alice and T be the input of Bob. For $1 \leq i \leq k$ define $A_i \stackrel{\text{def}}{=} S_i \setminus \cup_{i \neq j} S_j$; that is, A_i contains the elements that appear only in S_i. Let A_j be a smallest set among $A_1, \ldots, A_k$; that is, $|A_j| \leq |A_i|$ for $1 \leq i \leq k$. Since $A_1, \ldots, A_k$ are disjoint, $|A_j| \leq n/k$. To solve $\texttt{r-eliminate}_{\mathrm{DISJ}^k}$, Alice sends the set A_j to Bob. Bob computes $\mathrm{DISJ}(A_j, T)$ and sends the answer to Alice. Alice computes the output as follows: If $\mathrm{DISJ}(A_j, T) = 0$, then A_j intersects T, and, in particular, S_j intersects T. Therefore, in this case, Alice may return any vector whose j-th coordinate is 1. If $\mathrm{DISJ}(A_j, T) = 1$, then either S_j and T are disjoint, or $S_j \setminus A_j$ intersects T, and therefore S_i intersects T for at least one $i \neq j$ (since any element in $S_j \setminus A_j$ belongs to some S_i). Therefore, the vector whose j-th coordinate is 0 and all other coordinates are 1 is not possible and Alice outputs this vector.

The number of sets of size at most n/k is at most $(ek)^{n/k}$. Therefore, communicating the set A_j requires at most $\frac{n}{k} \log(ek) \leq (n/k)(\log k + 2)$ bits. □

Ambainis et al. [2] showed that $\mathcal{D}(\texttt{eliminate}_{\mathrm{DISJ}^k}) = \Omega(n)$. Using the fact that DISJ with n/k-bit inputs is padable with respect to DISJ with n-bit inputs and Lemma 1, we can obtain lower bounds on the complexity of r-eliminate of disjointness, that is, $\mathcal{D}(\texttt{r-eliminate}_{\mathrm{DISJ}^k}) = \Omega(n/k)$. In particular, we deduce that the protocol of Theorem 6 is optimal up to factor of $\log k$ for deterministic protocols. Theorem 5 and Lemma 1 imply the lower bound $\mathcal{R}_\epsilon(\text{r-}\texttt{eliminate}_{\mathrm{DISJ}^k}) = n/2^{O(k)}$ for $\epsilon < 1/2^{k+1}$.

R-Eliminate for the Inner Product function. Let $\mathrm{IP} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be the inner product mod 2 function, i.e. $\mathrm{IP}(x,y) = x[1] \cdot y[1] \oplus, \ldots, \oplus x[n] \cdot y[n]$, where $x[i]$ and $y[i]$ are the i-th bits of x and y respectively. In the full version of this paper, we show that some small saving is possible for $\texttt{r-eliminate}$ of IP; namely, $\mathcal{D}(\texttt{r-eliminate}_{\mathrm{IP}^k}) \leq \max\{n - k + 2, 0\}$. We next prove a lower bound on the randomized communication complexity of $\texttt{eliminate}_{\mathrm{IP}^k}$. Specifically, we prove that $\mathcal{R}_\delta(\texttt{eliminate}_{\mathrm{IP}^k}) \geq n - O(k)$, for $\delta < \frac{1}{4k \cdot 2^{2k}}$. This improves over a lower bound of $n/2^{O(k)} \log n \log\log n$ from [2]. Notice that for $k \geq \log n$ their

bound is $\Omega(1)$, while even for $k = o(n)$ our bound is $\Omega(n)$. The lower bound for IP can also be obtained from Theorem 5; we present the proof below since we believe that its ideas might be of interest.

Theorem 7. *$\mathcal{R}_\delta(\mathtt{eliminate}_{\mathrm{IP}^k}) \geq n - O(k)$, for $\delta < 1/(4k^{3/2}2^k + 2)$.*

Proof. We will show that if there is a δ-error protocol $\mathcal{P}$ for $\mathtt{eliminate}_{\mathrm{IP}^k}$ with complexity less than $n - O(k)$, then there is a randomized protocol for IP on inputs of length nk with error less than $1/2 - \epsilon$ with complexity less than $nk - \log O(1/\epsilon)$ contradicting the known lower bound for IP. Given $x, y \in \{0,1\}^{nk}$, the protocol for $\mathrm{IP}(x, y)$, denoted $\mathcal{P}'$, proceeds as follows:

1. Let $x = x_1, \ldots, x_k$ and $y = y_1, \ldots, y_k$, where $|x_i| = |y_i| = |n|$. Alice and Bob execute the protocol $\mathcal{P}$ for $\mathtt{eliminate}_{\mathrm{IP}^k}$ on $(x_1, \ldots, x_k, y_1, \ldots, y_k)$. Let $(\sigma_1, \ldots, \sigma_k)$ be the output of $\mathcal{P}$. Denote $\alpha_i = \overline{\sigma_i}$ for $1 \leq i \leq k$ (with probability at least $1 - \delta$, $\mathrm{IP}(x_i, y_i) = \alpha_i$ for at least one i).
2. Alice chooses uniformly at random an index $1 \leq j \leq k$ and sends j and $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_k$ to Bob.
3. Bob computes $\beta_i = \mathrm{IP}(x_i, y_i)$, for $i \in \{1, \ldots, k\} \setminus \{j\}$, computes $a = \oplus_{i \neq j} \beta_i$, and $c = |\{i \neq j : \alpha_i = \beta_i\}|$. It computes the protocol's output as follows:
   - if $c = 0$ (that is, $\alpha_i \neq \beta_i$ for every $i \in \{1, \ldots, k\} \setminus \{j\}$), the output is $a \oplus \alpha_j$ (in this case if $\mathcal{P}$ returns a correct output, then it must be that $\alpha_j = \beta_j$ and the output of $\mathcal{P}'$ is correct).
   - if $c > 0$, then with probability $1/2 + \epsilon_c$ the output is $a \oplus \alpha_j$ and with probability $1/2 - \epsilon_c$ it is $a \oplus \overline{\alpha_j}$, where ϵ_c will be determined later.

We analyze the success probability of the protocol $\mathcal{P}'$. First, assume that $\mathcal{P}$ never errs. We will later remove this assumption. Let $m = |\{i : \alpha_i = \mathrm{IP}(x_i, y_i)\}|$; that is, m is the number of correct values among $\alpha_1, \ldots, \alpha_k$.

The case $m = 1$: If Alice chooses the unique j such that $\alpha_j = \mathrm{IP}(x_j, y_j)$, then $c = 0$ and $\mathcal{P}'$ always succeeds. If Alice chooses any other j, then $\alpha_j \neq \mathrm{IP}(x_j, y_j)$ and $c = 1$, and so the protocol $\mathcal{P}'$ succeeds with probability $1/2 - \epsilon_1$. All together, the success probability, in this case, is:

$$\frac{1}{k} + \frac{k-1}{k}\left(\frac{1}{2} - \epsilon_1\right) \geq \frac{1}{2} + \frac{1}{2k} - \epsilon_1.$$

The case $2 \leq m \leq k$: If Alice chooses an index j such that $\alpha_j = \mathrm{IP}(x_j, y_j)$ (this happens with probability m/k), then $c = m - 1$ and Bob outputs the correct value (i.e., $a \oplus \alpha_j$) with probability $1/2 + \epsilon_{m-1}$. If Alice chooses j such that $\alpha_j \neq \mathrm{IP}(x_j, y_j)$ (with probability $(1 - m/k)$), then $c = m$ and Bob outputs the correct value (i.e., $a \oplus \overline{\alpha}_j$) with probability $1/2 - \epsilon_m$. All together, the success probability, in this case, is

$$\frac{m}{k}\left(\frac{1}{2} + \epsilon_{m-1}\right) + \left(1 - \frac{m}{k}\right)\left(\frac{1}{2} - \epsilon_m\right) = \frac{1}{2} + \frac{m}{k}\epsilon_{m-1} - \frac{k-m}{k}\epsilon_m.$$

Set $\epsilon_m = 1/(4\binom{k}{m})$. Thus, for $2 \leq m \leq k$, the success probability is:

$$\frac{1}{2} + \frac{1}{k}\left(\frac{m}{4\binom{k}{m-1}} - \frac{k-m}{4\binom{k}{m}}\right) = \frac{1}{2} + \frac{1}{4k\binom{k}{m}}.$$

For $m = 1$ the success probability is greater than $1/2 + 1/(4k)$. All together, $\mathcal{P}'$ succeeds with probability greater than $1/2 + \frac{1}{4k\binom{k}{k/2}} \geq 1/2 + 1/(4k^{3/2}2^k)$ (since $\binom{k}{k/2} \leq 2^k/k^{1/2}$).

Next, assume that $\mathcal{P}$ errs with probability (at most) δ. In the worst case, $\mathcal{P}'$ fails whenever $\mathcal{P}$ fails. The success probability of $\mathcal{P}'$ is, therefore, at least $(1-\delta)\cdot(1/2 + 1/(4k^{3/2}2^k))$. Assuming that $\delta \leq 1/(4k^{3/2}2^k + 2)$, the success probability of $\mathcal{P}'$ is at least $1/2 + 1/(8k2^{2k})$.

The communication complexity of $\mathcal{P}'$ is the communication complexity of $\mathcal{P}$, on inputs of length n, plus $(1-1/k)nk$. By [21, Exercise 3.30], the communication complexity of IP with error $1/2 - \epsilon$ on inputs of length nk is at least $nk - O(\log 1/\epsilon)$. Thus, we get $\mathcal{R}_\delta(\texttt{eliminate}_{\mathrm{IP}^k}) + (1-1/k)\cdot nk \geq nk - O(k)$, which implies that $\mathcal{R}_\delta(\texttt{eliminate}_{\mathrm{IP}^k}) \geq n - O(k)$. □

As IP with n/k-bit inputs is padable with respect to IP with n-bit inputs (using $b = 0$ in Definition 3) then, using Lemma 1, we get:

Corollary 1. $\mathcal{R}_\delta(\text{r-}\texttt{eliminate}_{\mathrm{IP}^k}) \geq \frac{n}{k} - O(1)$ *for every* $\delta < 1/(4k^{3/2}2^k + 2)$.

We know that $\mathcal{R}_{1/2^k}(\texttt{eliminate}_{\mathrm{IP}^k}) = O(1)$. Thus, the error that we allow in Theorem 7 (and Corollary 1) is nearly optimal.

Eliminate of Most Functions. In the full version of the paper, we prove that for most functions f `r-eliminate` cannot be computed efficiently; i.e., we prove that $\mathcal{D}(\texttt{r-eliminate}_{f^2}) \geq n - 5$ for most functions.

References

1. Agrawal, M., Arvind, V.: Polynomial time truth-table reductions to P-selective sets. In: Structure in Complexity Theory Conference, pp. 24–30 (1994)
2. Ambainis, A., Buhrman, H., Gasarch, W., Kalyanasundaramd, B., Torenvliete, L.: The communication complexity of enumeration, elimination, and selection. JCSS 63(2), 148–185 (2001)
3. Amihood, A., Beigel, R., Gasarch, W.I.: Some connections between bounded query classes and non-uniform complexity. ECCC 7(024) (2000)
4. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: An information statistics approach to data stream and communication complexity. In: Proceedings of the 43rd Symposium on Foundations of Computer Science, pp. 209–218 (2002)
5. Beigel, R., Gasarch, W.I., Gill, J., Owings, J.C.: Terse, superterse, and verbose sets. Inf. Comput. 103(1), 68–85 (1993)
6. Beigel, R., Gasarch, W.I., Kummer, M., Martin, G., McNicholl, T., Stephan, F.: The complexity of Odd_n^A. J. Symb. Log. 65(1), 1–18 (2000)

7. Beigel, R., Hirst, T.: One help-bit doesn't help. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing, pp. 124–130 (1998)
8. Beigel, R., Kummer, M., Stephan, F.: Approximable sets. Information and Computation 120, 12–23 (1994)
9. Cai, J., Hemachandra, L.A.: Enumerative counting is hard. Inf. Comput. 82(1), 34–44 (1989)
10. Feder, T., Kushilevitz, E., Naor, M., Nisan, N.: Amortized communication complexity. SIAM Journal on Computing 24(4), 736–750 (1995)
11. Galbiati, G., Fischer, M.J.: On the complexity of 2-output Boolean networks. Theor. Comput. Sci. 16, 177–185 (1981)
12. Gasarch, W., Martin, G.: Bounded queries in recursion theory (1998)
13. Hemaspaandra, L.A., Torenvliet, L.: Theory of Semi-Feasible Algorithms (2003)
14. Jain, R., Klauck, H., Santha, M.: Optimal direct sum results for deterministic and randomized decision tree complexity. Technical Report 1004.0105v1, arxiv.org (2010), http://arxiv.org/abs/1004.0105v1
15. Jockusch, C.: Semirecursive sets and positive reducibility. Transactions of the American Mathematical Society 131(2), 420–436 (1968)
16. Karchmer, M., Kushilevitz, E., Nisan, N.: Fractional covers and communication complexity. SIAM J. on Discrete Mathematics 8(1), 76–92 (1995)
17. Karchmer, M., Raz, R., Wigderson, A.: On proving superlogarithmic depth lower bounds via the direct sum in communication complexity. In: 6th IEEE Structure in Complexity Theory, pp. 299–304 (1991)
18. Karchmer, M., Wigderson, A.: Monotone circuits for connectivity require superlogarithmic depth. SIAM J. on Discrete Mathematics 3(2), 255–265 (1990)
19. Ko, K.: On self-reducibility and weak P-selectivity. JCSS 26(2), 209–221 (1983)
20. Kummer, M.: A proof of Beigel's cardinality conjecture. J. Symb. Log. 57(2), 677–681 (1992)
21. Kushilevitz, E., Nisan, N.: Communication Complexity (1997)
22. Nisan, N., Rudich, S., Saks, M.: Products and help bits in decision trees. SIAM J. Comput. 28(3), 1035–1050 (1999)
23. Paul, W.J.: Realizing Boolean functions on disjoint sets of variables. Technical report, Cornell University, Ithaca, NY, USA (1974)
24. Selman, A.L.: P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. In: 6th ICALP, pp. 546–555. Springer, Heidelberg (1979)
25. Selman, A.L.: Analogues of semicursive sets and effective reducibilities to the study of NP complexity. Information and Control 52(1), 36–51 (1982)
26. Selman, A.L.: Reductions on NP and P-selective sets. Theor. Comput. Sci. 19, 287–304 (1982)
27. Shaltiel, R.: Towards proving strong direct product theorems. In: Proc. of the 16th IEEE Conf. on Computational Complexity, pp. 107–119 (2001)
28. Sivakumar, D.: On membership comparable sets. J. Comput. Syst. Sci. 59(2), 270–280 (1999)
29. Uhlig, D.: On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements. Mat. Zametki 15, 937–944 (1974)
30. Yao, A.C.: Some complexity questions related to distributed computing. In: Proc. of the 11th ACM Symp. on the Theory of Computing, pp. 209–213 (1979)

Additive Spanners in Nearly Quadratic Time

David P. Woodruff

IBM Almaden

Abstract. We consider the problem of efficiently finding an additive C-spanner of an undirected unweighted graph G, that is, a subgraph H so that for all pairs of vertices u, v, $\delta_H(u,v) \leq \delta_G(u,v) + C$, where δ denotes shortest path distance. It is known that for every graph G, one can find an additive 6-spanner with $O(n^{4/3})$ edges in $O(mn^{2/3})$ time. It is unknown if there exists a constant C and an additive C-spanner with $o(n^{4/3})$ edges. Moreover, for $C \leq 5$ all known constructions require $\Omega(n^{3/2})$ edges.

We give a significantly more efficient construction of an additive 6-spanner. The number of edges in our spanner is $n^{4/3}$polylog n, matching what was previously known up to a polylogarithmic factor, but we greatly improve the time for construction, from $O(mn^{2/3})$ to n^2polylog n. Notice that $mn^{2/3} \leq n^2$ only if $m \leq n^{4/3}$, but in this case G itself is a sparse spanner. We thus provide both the fastest and the sparsest (up to logarithmic factors) known construction of a spanner with constant additive distortion.

We give similar improvements in the construction time of additive spanners under the assumption that the input graph has large girth, or more generally, the input graph has few edges on short cycles.

1 Introduction

An additive C-spanner (often called a $(1, C)$-spanner) of an unweighted undirected graph G is a subgraph H with the property that for all vertices u, v, $\delta_H(u,v) \leq \delta_G(u,v) + C$. Here C is known as the distortion.

Spanners have a variety of applications. They are used in space-efficient routing tables that guarantee almost shortest routes [1], [10], [11], [17], [20], methods for simulating synchronized protocols in unsynchronized networks [16], parallel and distributed algorithms for computing almost shortest paths [7], [8], [14], and in labeling schemes and distance oracles for reporting approximate distances [4], [6], [19], [21].

There is a tradeoff between the distortion and the sparsity of the spanner. Studying what sparsity is possible for small constant C has received considerable attention. For $C = 2$, Aingworth *et al* (see [2], and the followup work [5], [12], [15], [18], [22]) show that $O(n^{3/2})$ edges are necessary and sufficient; that is, every graph G contains an additive 2-spanner with this many edges, and there exist graphs for which any additive 2-spanner requires this many edges. Later, Baswana *et al* [5] show that every graph G contains an additive 6-spanner with $O(n^{4/3})$ edges. Nothing better than $O(n^{4/3})$ is known even if the distortion is

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 463–474, 2010.

allowed to be any constant, though lower bounds that degrade with the distortion are known [23].

Another important measure is the time complexity needed for constructing such a spanner. As many graph algorithms for distance-approximation [2], [5], [9], [12] have running time proportional to the number of edges, a key approach is to first find a spanner of the input graph, and then run existing algorithms on the spanner rather than the dense input. The value of this approach is diminished if the time to find the spanner is already too large.

The additive 2-spanner of Aingworth *et al* has construction time $O(mn^{1/2})$. This was improved by Dor, Halpern, and Zwick [12] to $\tilde{O}(n^2)$ time[1] and $\tilde{O}(n^{3/2})$ edges. The conference version of the additive 6-spanner construction of Baswana *et al* [5] had $O(mn)$ time, but was improved to $O(mn^{2/3})$ in the journal version [3], and the improvement is attributed to Elkin [13]. Notice that $O(mn^{2/3})$ is much larger than $\tilde{O}(n^2)$. Indeed, $m > n^{4/3}$; otherwise G itself serves as a sparse spanner. Hence, $mn^{2/3} > n^2$.

It should be noted that prior to the work of Baswana *et al* [5], Dor, Halpern, and Zwick [12] constructed what they called a 6-emulator of a graph with $\tilde{O}(n^{4/3})$ edges and $\tilde{O}(n^2)$ time. A 6-emulator H of a graph $G = (V, E)$ is a weighted graph on the same vertex set V with an arbitrary edge set E', subject to the constraint that $d_G(u, v) \leq d_H(u, v) \leq d_G(u, v) + 6$. Emulators are often much easier to construct than spanners. For instance, it is known that every graph has a 4-emulator containing $\tilde{O}(n^{4/3})$ edges [12], but the best known additive 4-spanner has size $\Theta(n^{3/2})$. It was not until the much later work of [5] that an additive 6-spanner with $O(n^{4/3})$ edges was constructed, albeit with a much larger running time.

1.1 Results

Our main contribution is a new construction of an additive 6-spanner with $\tilde{O}(n^{4/3})$ edges in a significantly faster $\tilde{O}(n^2)$ time. We thus provide both the fastest and the sparsest (up to logarithmic factors) known construction of a spanner with constant additive distortion.

Our techniques also solve the following problem: given a subset S of $O(n^{2/3})$ vertices of a graph G, find a subgraph H of G containing $\tilde{O}(n^{4/3})$ edges so that for all $u, v \in S$, $\delta_H(u, v) \leq \delta_G(u, v)+2$. Our method solves this problem in $\tilde{O}(n^2)$ time. The best previous time was $O(mn^{2/3})$, using a technique of [3], attributed to Elkin.

In an attempt to achieve even sparser additive spanners, Baswana *et al* [5] study the construction of spanners by parameterizing the number of edges appearing on any short cycle. Let $\Gamma_k(G)$ be the number of edges in G that lie on a cycle with length at most $2k$. They show that for any integers $k \geq 1$ and $\ell \in [0, k]$, there exists an additive $(2k + 4\ell)$-spanner with $O(\Gamma_k(G) + n^{1+\frac{1}{k+\ell+1}})$ edges that can be constructed in $O(mn^{1-\frac{\ell}{k+\ell+1}})$ time. We show that an additive $(2k + 4\ell)$-spanner with $\tilde{O}(\Gamma_k(G) + n^{1+\frac{1}{k+\ell+1}})$ edges can be constructed in $\tilde{O}(n^2)$

[1] Let $\tilde{O}(f(n))$ denote the class of functions that are bounded above by a function that grows as $f(n) \cdot \text{polylog}(f(n))$.

time. This gives a sizable improvement upon the previous time bound for a wide range of k and ℓ.

As an example setting of parameters, if the input graph has girth greater than 4, we can construct an additive 4-spanner (resp. additive 8-spanner) with $\tilde{O}(n^{4/3})$ edges (resp. $\tilde{O}(n^{5/4})$ edges) in $\tilde{O}(n^2)$ time, whereas the prior bound was $O(mn)$ time (resp. $\tilde{O}(mn^{3/4})$) time.

1.2 Techniques

Our method is based on a new path-hitting framework. This approach is quite different than the path-purchasing methods used in [5] to construct an additive 6-spanner. Namely, our work is the first to look at hitting the neighborhood of a path in the absence of high-degree vertices, provided there are enough vertices of moderate degree along the path.

In more detail, as in [3], we initially include all edges incident on low-degree ($< n^{1/3}$) vertices in the spanner. We then find a dominating set D_1 of size $\tilde{O}(n^{2/3})$ of all vertices of degree at least $n^{1/3}$. Each vertex in D_1 can be thought of as a cluster center, as in [3], and for each vertex incident to at least one cluster center, we include an edge to one such center in our spanner. Now our algorithm deviates from that of [3]. That algorithm needs to compute a breadth-first search (BFS) tree around each cluster center. It then uses these trees to iteratively choose walks (i.e., paths with repeated vertices and edges) to include in the spanner. The main problem is the BFS tree computations, taking $O(mn^{2/3})$ time, which is too costly. We now explain how to overcome this problem.

We first describe a new combinatorial way of looking at the problem. Suppose each shortest path P has x edges missing from the spanner. Then there are $\Omega(n^{1/3}x)$ vertices which are at distance 1 from some vertex in P. This follows from the fact that the neighborhoods of two vertices that are at a distance larger than 2 on P cannot intersect, by the triangle inequality. Therefore, if we choose a random sample D_2 of $\tilde{O}(n^{2/3}/x)$ of the n vertices, we will hit the neighborhood of each shortest path. If $u \in D_2$ is in the neighborhood of P, and $v_1, v_r \in D_1$ are neighbors of the first and last missing edge in P, then there are almost shortest walks from u to v_1 and u to v_r with a total of roughly x missing edges. If we include the missing edges on these walks to the spanner for all pairs of vertices in D_1 and D_2, the total number of edges added is $\tilde{O}((n^{2/3}/x) \cdot n^{2/3} \cdot x) = \tilde{O}(n^{4/3})$. Since x is not the same for all shortest paths, we need to repeat this procedure for (a logarithmic number of) geometrically increasing intervals of x.

There are several technical hurdles we need to overcome to implement this in $\tilde{O}(n^2)$ time. First, we need a procedure, which, given a subgraph of the input, finds almost shortest walks between pairs of vertices with the smallest number of missing edges efficiently. For this bicriteria problem, we design a procedure FHEASW-Solver which runs in $\tilde{O}(m')$ time, where m' is the number of edges in the subgraph. This procedure is based on a BFS-like dynamic program. We can then run FHEASW-Solver from each vertex in D_2, provided for the given value

of x and the shortest paths P with x edges missing, we can find a subgraph which contains P and that has $\tilde{O}(n^{4/3}x)$ edges. Indeed, in this case the time of the invocations of FHEASW-Solver will be $\tilde{O}(|D_2|n^{4/3}x) = \tilde{O}(n^2)$. If we choose the subgraph to be all edges with one endpoint of degree at most $n^{1/3}x$, then it has at most $\tilde{O}(n^{4/3}x)$ edges, as we need. Unfortunately this subgraph may not contain P if P contains a vertex of degree larger than $n^{1/3}x$.

This leads us to our final case. If P contains a vertex of degree larger than $n^{1/3}x$, then we can compute another dominating set D_3 of all vertices of degree at least d in G, where d is the largest degree of a vertex along P. This dominating set has size $\tilde{O}(n/d)$. We can then connect each vertex in this dominating set to each of the vertices in D_1 via an almost shortest walk missing at most $d/n^{1/3}$ edges. Indeed, by assumption $d > n^{1/3}x$, or equivalently, the number x of missing edges is at most $d/n^{1/3}$. The total number of edges added is therefore $\tilde{O}(|D_1|(n/d)(d/n^{1/3})) = \tilde{O}(n^{4/3})$. To find the edges to add in this step we need to run yet another invocation of FHEASW-Solver. We run it on a subgraph of $O(nd)$ edges, once from each vertex in D_3, therefore taking $\tilde{O}(|D_3|nd) = \tilde{O}(n^2)$ total time. Since d is not the same for all shortest paths, we need to vary it in geometrically increasing intervals as well. This idea of varying the degree and working in subgraphs is similar to ideas used for additive 2-spanners of Dor, Halpern, and Zwick [12].

An analysis of the union of the edgesets from the two cases shows that for any pair of vertices, there is a path in the spanner between them with distortion at most 6. It turns out that achieving this guarantee requires running FHEASW-Solver several times on the same subgraph with different parameters. We remark that our algorithm is Monte Carlo, that is, with high probability an additive-6 spanner is returned. We do not know how to derandomize the algorithm in $\tilde{O}(n^2)$ time due to the fact that we need to hit the neighborhood of many paths, but can only implicitly represent the paths in the allotted $\tilde{O}(n^2)$ time.

2 Preliminaries

The input is an unweighted undirected graph $G = (V, E)$ on a set V of n vertices and a set E of m edges. W.l.o.g., n is a power of 8. Let $\deg(u)$ be the degree of $u \in V$, and let $N(u)$ be the set of neighbors of u. For a set S of vertices, let $N(S)$ be $\cup_{u \in S} N(u)$.

Suppose E' is an arbitrary subset of E. For $u \in V$, let $\mathrm{BFS}(u, (V, E'))$ denote a shortest-path tree rooted at u in the graph (V, E'). The following theorem is standard and stated here for reference. See, e.g., the discussion in [9] or [12].

Theorem 1. *(BFS) Given an unweighted undirected graph $G = (V, E)$ and a vertex $u \in V$, there is an $O(m + n)$-time algorithm that finds distances, and a tree of shortest paths, from u to all the vertices of V.*

All logarithms, if unspecified, are to the base 2. We will classify all edges of our input graph G according to their type: an edge $e = \{u, v\}$ is of type i if $\min(\deg(u), \deg(v)) \in [2^i, 2^{i+1})$. An edge is *light* if it is of type i for an

$i \leq \log n^{1/3}$. Otherwise, it is *heavy*. We assume the standard RAM model on $O(\log n)$-sized words in which arithmetic operations take $O(1)$ time.

3 Constructing Additive 6-Spanners

3.1 A Subroutine

In our main algorithm, we need a subroutine which solves the following problem. Fix an unweighted undirected graph $G = (V, E)$. Fix a root vertex $u \in V$. For each vertex $v \in V$, suppose we know $\delta_G(u, v)$. We would like to find a walk from u to v with the least number of heavy edges among all walks from u to v whose distance is at most $\delta_G(u, v) + C$ for a constant $C > 0$. More precisely, for a walk P let $\phi(P)$ be the number of heavy edges along P. We sometimes abuse notation and let $\phi\{u, v\} = \phi(\{u, v\})$.

Fewest Heavy Edges with Almost Shortest Walks (FHEASW): Given G, a vertex u, and a constant C, output a data structure for which on input $v \in V$ and $i \in \{\delta_G(u, v), \delta_G(u, v) + 1, \ldots, \delta_G(u, v) + C\}$, there is an algorithm that returns, in $O(\phi(P))$ time, the heavy edges along a walk P from u to v in G with $\delta_P(u, v) = i$ and $\phi(P) = \min_{\text{walks } P' \mid \delta_{P'}(u,v) = i} \phi(P')$.

Note that we allow P to be a walk rather than just a path, that is, it is a path that may contain repeated vertices and edges.

We can solve this problem as follows. For a $v \in V$ and $i \in \{\delta_G(u, v), \delta_G(u, v) + 1, \ldots, \delta_G(u, v) + C\}$, let $D(v, i)$ be the minimum number of heavy edges on a walk to u with length $\delta_G(u, v)+i$. If there is no such walk, we define $D(v, i) = \infty$. Let $S(j)$ be the set of vertices w reachable by a walk from u of length exactly j, and for which $j \leq \delta_G(u, w) + C$. Then,

$$D(v, i) = \min_{w \in S(\delta_G(u,v)+i-1) \cap N(v)} D(w, i - 1) + \phi(w, v) \tag{1}$$

We can build up the values $D(v, i)$ as follows. We first obtain $\delta_G(u, v)$ for all v by running the algorithm of Theorem 1. Then to create $S(j)$ given $S(j - 1)$, we include each $v \in N(S(j - 1))$ for which $j \leq \delta_G(u, v) + C$. The time to do this is $O(|N(S(j - 1))|)$. When computing $N(S(j - 1))$, we can also update the appropriate $D(v, i)$ values by Equation (1). Hence, the total time is, up to a constant factor, $\sum_j |N(S(j - 1))|$, where $S(0) = \{u\}$. The key point is that, since C is constant, any given vertex can occur in at most $C + 1$ different sets $S(j)$. It follows that the total time is $O(Cm) = O(m)$.

While we have shown how to calculate the costs $D(v, i)$, to solve the FHEASW problem we must also return all heavy edges along a walk from u to v with length $\delta_G(u, v) + i$ and containing at most $D(v, i)$ heavy edges. One can do this by keeping side information in the algorithm above. Namely, each time $D(v, i)$ increases, we can append the heavy edge to a running list of heavy edges along the walk. This does not affect the overall time complexity by more than a constant factor. Due to space constraints, we defer further details to the full version of this paper.

3.2 Main Algorithm

Spanner Construction($G = (V, E)$):

1. Compute the type of each edge and store these values.
2. Initialize F to the set of light edges.
3. Repeat the following steps $3 \log n$ times:
   (a) Let R be a random sample of $2n^{2/3}$ vertices. For each vertex in $[n] \setminus R$, if there is an edge to a vertex in R, add one such edge to F.
   (b) For $i = \log n, (\log n) - 1, (\log n) - 2, \ldots, (\log n^{1/3}) + 1$,
      i. Let E' be the subset of E of edges of type at most i.
      ii. For $j = 0, 1, 2, \ldots, \lceil \log 3n/2^i \rceil$,
         A. Let $S_{i,j}$ be a random subset of $\lceil 3n/2^{i+j} \rceil$ vertices.
         B. For $u \in S_{i,j}$, $D_u = \text{FHEASW-Solver}((V, E'), u, 4)$.
         C. For $u \in S_{i,j}$, $v \in R$, $z \in \{\delta_{(V,E')}(u, v), \delta_{(V,E')}(u, v) + 1, \ldots, \delta_{(V,E')}(u, v) + 4\}$, add to F the heavy edges on the walk from u to v given by D_u with query input z, provided the number of heavy edges on the walk is $\leq 2^{i+j+1}/n^{1/3} + 2$.
4. Output $H = (V, F)$.

Theorem 2. $|F| = O(n^{4/3} \log^3 n)$.

Proof. The number of light edges added to F is $O(n^{4/3})$. The number of edges added in step 3a, over all $3 \log n$ iterations of step 3, is $O(n \log n)$.

For each iteration i of step 3b, at most $\sum_j |R||S_{i,j}|(2^{i+j+1}/n^{1/3} + 2) = \sum_j O(n^{2/3}) \cdot O(n/2^{i+j}) \cdot (2^{i+j+1}/n^{1/3} + 2) = O(n^{4/3} \log n) + \sum_j O(n^{5/3})/2^{i+j} = O(n^{4/3} \log n)$ edges are added to F, where the last inequality follows from the fact that $2^{i+j} \geq n^{1/3}$. The number of iterations of step 3b is $O(\log n)$, and step 3 is invoked $3 \log n$ times, resulting in $|F| = O(n^{4/3} \log^3 n)$.

Theorem 3. Spanner Construction *can be implemented in* $O(n^2 \log^2 n)$ *time.*

Proof. In step 1 we classify each edge as light or heavy, and assign the corresponding weight. This can be done in $O(m)$ time.

A single iteration of step 3a takes at most $O(m)$ time. Hence, over all $O(\log n)$ iterations, step 3a takes $O(m \log n) = O(n^2 \log n)$ time.

For a single iteration of step 3biiB, for a fixed value of i, j, and $u \in S_{i,j}$, FHEASW-Solver takes time $O(n2^i)$, since there are at most $O(n2^i)$ edges in the subgraph. As there are $O(n/2^{i+j})$ different $u \in S_{i,j}$, it follows that step 3biiB takes $O(n^2/2^j)$ time. Hence, summing over all iterations and i and j, step 3biiB takes $O(n^2 \log^2 n)$ time.

A single iteration of step 3biiC takes at most $O(|S_{i,j}||R|(2^{i+j+1}/n^{1/3} + 2)) = O(n/2^{i+j}) \cdot O(n^{2/3})(2^{i+j+1}/n^{1/3} + 2) = O(n^{4/3})$ time. Summing over iterations, step 3biiC takes $\tilde{O}(n^{4/3})$ time.

Hence, the total time of the algorithm is $O(n^2 \log^2 n)$.

It remains to argue the algorithm's correctness. We start with a lemma. If $P = (v_1, \ldots, v_s)$ is a shortest path from vertex v_1 to v_s, we define $N(P) = \cup_{i=1}^{s} N(v_i)$.

Lemma 1. *In the graph (V, E') for any $E' \subseteq E$, if there is a shortest path P from u to v containing ℓ vertices of degree at least $2n^{1/3}$, then $|N(P)| \geq 2\ell n^{1/3}/3$.*

Proof. Let $w_1, w_2, \ldots, w_\ell$ be the sequence of vertices of degree at least $2n^{1/3}$ along P (possibly with other vertices in between). Observe that for each j, $N(w_j)$ must be disjoint from $\cup_{j' \geq j+3} N(w_{j'})$. Otherwise one could go from w_j to a vertex $x \in N(w_j) \cap (\cup_{j' \geq j+3} N(w_{j'}))$, then to a vertex $w_{j'}$ for some $j' \geq j+3$, in two steps. As $\delta_P(w_j, w_{j'}) \geq 3$, this contradicts P being a shortest path, since each of its sub-paths must be shortest. As $|N(w_j)| \geq 2n^{1/3}$ for all j, it follows that $\left|\cup_{j=1}^{\ell} N(w_j)\right| \geq 2\ell n^{1/3}/3$.

Theorem 4. *With probability at least $1 - 1/n$, H is an additive 6-spanner.*

Proof. Fix a pair $\{a, b\}$ of vertices in G, and fix any shortest path P from a to b in G with the fewest heavy edges. We assume that there is at least one heavy edge on P, as otherwise the path P will be added to F in step 2. So there are at least two vertices of degree at least $2n^{1/3}$ on P. Let $w_1, w_2, \ldots, w_r$ be the ordered sequence of vertices on P of degree at least $2n^{1/3}$, where w_1 is closest to a and w_r is closest to b.

Consider one iteration of step 3. We show that with probability at least 3/8, using only the edges added to F in step 2 and the current iteration of step 3, there is a path of length at most $\delta_G(a, b) + 6$ from a to b. It will follow that the probability that there is some iteration for which there is a path of length at most $\delta_G(a, b) + 6$ is at least $1 - (3/8)^{3 \log n} = 1 - 1/n^3$. By a union bound, it will follow that for every pair of vertices, there is such a path with probability at least $1 - 1/n$, that is, H is an additive 6-spanner.

Let i^* be such that all edges along P have type at most i^*, and there is at least one edge e^* of type i^*. Observe that $i^* \in \{\log n, (\log n) - 1, (\log n) - 2, \ldots, (\log n^{1/3}) + 1\}$. We shall only consider the i^*-th iteration of step 3b. Notice that the entire path P is included in the edgeset E'. We do a case analysis.

Case 1: The path P contains at most $2^{i^*}/n^{1/3}$ heavy edges.

In this case we shall only consider the iteration in which $j = 0$ of step 3bii. Now, e^* is of type i^*, which means that at least one of its endpoints y has degree in the range $[2^{i^*}, 2^{i^*+1})$. It follows that all edges incident to y have type at most i^*, and in particular, are included in the graph (V, E'). Consider the following event $\mathcal{E}$: $S_{i^*,0} \cap N(y) \neq \emptyset$. Then,

$$\Pr[\mathcal{E}] \geq 1 - \left(1 - \frac{|N(y)|}{n}\right)^{|S_{i^*,0}|} \geq 1 - \left(1 - \frac{2^{i^*}}{n}\right)^{3n/2^{i^*}} \geq 1 - \frac{1}{e}.$$

Conditioned on $\mathcal{E}$, let $u \in S_{i^*,0} \cap N(y)$. Let E^* be the set of edges added in step 3a, and consider the event $\mathcal{F}$: $\exists v_1 \in R$ for which $\{w_1, v_1\} \in E^*$ and $\exists v_r \in R$ for which $\{w_r, v_r\} \in E^*$. Since $\text{degree}(w_1), \text{degree}(w_r) \geq 2n^{1/3}$, $\Pr[\mathcal{F}] \geq 1 - 2\left(1 - \frac{2n^{1/3}}{n}\right)^{2n^{2/3}} \geq 1 - 2e^{-4}$. By a union bound, $\Pr[\mathcal{E} \wedge \mathcal{F}] \geq 1 - \frac{1}{e} - \frac{2}{e^4} > \frac{3}{8}$.

Conditioned on $\mathcal{E} \wedge \mathcal{F}$, let v_1 be a vertex in R for which $\{w_1, v_1\} \in E^*$, and v_r be a vertex in R for which $\{w_r, v_r\} \in E^*$. Consider the walk Q from u to v_1 which first traverses edge $\{u, y\}$, then agrees with path P from y to w_1, then traverses edge $\{w_1, v_1\}$. Observe that the number of heavy edges along Q is at most $2^{i^*}/n^{1/3} + 2$, since P contains at most $2^{i^*}/n^{1/3}$ heavy edges. Moreover, by the triangle inequality, the walk Q is of length at most $\delta_G(u, v_1) + 4 \leq \delta_{(V,E')}(u, v_1) + 4$. It follows that in step 3biiC, there will be a walk Q' added to F from v_1 to u of length at most $\delta_{(V,E')}(w_1, y) + 2$. Similarly, there will be a walk Q'' added to F from u to v_r of length at most $\delta_{(V,E')}(y, w_r) + 2$.

Hence, the walk P' from a to b which agrees with P from a until w_1, then goes to v_1, then takes the walk Q' to u, then the walk Q'' from u to v_r, then goes to w_r, then agrees with P from w_r to b, is of length at most $|P'| \leq \delta_G(a, w_1) + 1 + |Q'| + |Q''| + 1 + \delta_G(w_r, b)$, which is at most $\delta_G(a, w_1) + 1 + \delta_{(V,E')}(w_1, y) + 2 + \delta_{(V,E')}(y, w_r) + 2 + 1 + \delta_G(w_r, b)$, which is at most $\delta_G(a, b) + 6$.

Case 2: The path P contains more than $2^{i^*}/n^{1/3}$ heavy edges. Let j^* be such that the number of heavy edges on P is in the interval $[2^{i^*+j^*}/n^{1/3}, 2^{i^*+j^*+1}/n^{1/3})$. Each heavy edge on P is of type at most i^*, and so one of the two endpoints must have degree in the range $[2n^{1/3}, 2^{i^*+1})$. It follows that all of the edges incident to this endpoint in G are included in the graph (V, E'). It follows that there are at least $2^{i^*+j^*-1}/n^{1/3}$ vertices of degree at least $2n^{1/3}$ on P in the graph (V, E'). By Lemma 1, $|N(P)|$ is therefore at least $2^{i^*+j^*}/3$. Notice that this is also at most n, and therefore $2^{j^*} \leq 3n/2^{i^*}$. Hence there is an iteration of step 3bii for which $j = j^*$. We only consider this iteration.

By Lemma 1, $|N(P)| \geq 2^{i^*+j^*}/3$. Consider the following event $\mathcal{E} : S_{i^*,j^*} \cap N(P) \neq \emptyset$. Then, $\Pr[\mathcal{E}] \geq 1 - \left(1 - \frac{|N(P)|}{n}\right)^{|S_{i^*,j^*}|} \geq 1 - \left(1 - \frac{2^{i^*+j^*}}{3n}\right)^{3n/2^{i^*+j^*}} \geq 1 - \frac{1}{e}$. Conditioned on $\mathcal{E}$, let $u \in S_{i^*j^*} \cap N(P)$ and let $y \in P$ be such that $\{u, y\} \in E'$. As in case 1, letting E^* be the set of edges added in step 3a, we have that with probablity at least $3/8$ event $\mathcal{E}$ occurs and there is a vertex $v_1 \in R$ for which $\{w_1, v_1\} \in E^*$, and a vertex $v_r \in R$ for which $\{w_r, v_r\} \in E^*$.

As in case 1, consider the walk Q from u to v_1, traversing edge $\{u, y\}$, agreeing with P from y to w_1, then traversing edge $\{w_1, v_1\}$. This walk contains at most $2^{i^*+j^*+1}/n^{1/3} + 2$ heavy edges and has length at most $\delta_{(V,E')}(u, v_1) + 4$, so in step 3biiC there will be a walk Q' added to F from v_1 to u of length at most $\delta_{(V,E')}(w_1, y) + 2$. And as before, there will be a walk Q'' added to F from u to v_r of length at most $\delta_{(V,E')}(y, w_r) + 2$. It follows that as in case 1 that there is a walk P' from a to b in the spanner with length at most $\delta_G(a, b) + 6$. This completes the proof.

4 Construction for Inputs with Large Girth

Given a graph G with m edges and n vertices, let $\Gamma_k(G)$ be the number of edges in G that lie on a cycle with length at most $2k$. Choose ε so that $n^{\varepsilon} = (k^2 n)^{1/(k+\ell+1)}$. We need the following lemma of Baswana *et al.*

Lemma 2. *([5]) There are clusterings $\mathcal{C}_\ell$ and $\mathcal{C}_k$ of vertices of G with the following properties: (1) for $i \in \{\ell, k\}$, each cluster $C \in \mathcal{C}_i$ is a rooted spanning tree with radius at most i, and (2) the number of clusters in $\mathcal{C}_i$ is $n^{1-i\varepsilon}$. The subgraph $H_{k,\ell}$ containing all such spanning trees as well as every edge incident to a vertex that does not appear in both clusterings $\mathcal{C}_\ell$ and $\mathcal{C}_k$ can be constructed in $O(m)$ time and satisfies $\mathbf{E}[|H_{k,\ell}|] = O(\Gamma_k(G) + n^{1+\varepsilon})$.*

Now we define a heavy edge to be an edge that does not appear in $H_{k,\ell}$. It is easy to see that Theorem FHEASW-Solver continues to work with this new definition of a heavy edge, that is, FHEASW-Solver solves the FHEASW problem (with respect to this new definition of heavy) in $O(m+n)$ time, for constant k and ℓ.

We let Base-Edges denote the algorithm guaranteed by Lemma 2. The following is our main algorithm.

InterClusterSpanner $(G = (V, E))$:

1. Initialize F to the output of Base-Edges(G). For $i \in \{k, \ell\}$, let V_i denote the set of centers of vertices in clusters in $\mathcal{C}_i$.
2. Repeat the following steps $3 \log n$ times:
   (a) For $i = \log n, (\log n) - 1, (\log n) - 2, \ldots, 1$,
      i. Let S_i be a random sample of $2n/2^i$ vertices. For each vertex in $[n] \setminus S_i$, if there is an edge to a vertex in S_i, add one edge to F.
      ii. Put $q = \lfloor \log n^{-k\varepsilon} 2^i/(2k) \rfloor$. For $j = q, q+1, \ldots, \lceil \log n^{1-k\varepsilon} \rceil$, sample a set $S_{i,j} \subseteq V_k$ of $\lceil n^{1-k\varepsilon} 2^{-j} \rceil$ vertices.
      iii. Let $E' \subseteq E$ be the set of edges of type at most i.
      iv. For each $u \in S_i \cup (\cup_{\text{all } j} S_{i,j})$,
         A. Compute $D_u = \text{FHEASW-Solver}((V, E'), u, \ell + k)$.
         B. For $v \in V_\ell$, $z \in \{\delta_{(V,E')}(u,v), \delta_{(V,E')}(u,v) + 1, \ldots, \delta_{(V,E')}(u,v) + 2\ell + 2k\}$, add to F the heavy edges along the walk from u to v given by D_u with query input z, provided it has at most $2^i n^{-k\varepsilon} + \ell + 1$ heavy edges if $u \in S_i$ and $2^i \geq n^{k\varepsilon}$, or at most $4k2^j + \ell + k$ heavy edges if $u \in S_{i,j}$ for some j.
3. Output $H = (V, F)$.

Our analysis will assume that the output of Base-Edges in step 1 has $O(\Gamma_k(G) + n^{1+\varepsilon})$ edges. This can be done with probability $1 - 1/n^3$ by running the algorithm of Lemma 2 $O(\log n)$ times and taking the output with the least number of edges.

Theorem 5. $|F| = O(\Gamma_k(G) + n^{1+\frac{1}{k+\ell+1}} \log^3 n)$.

Proof. The number of edges added in step 1 is $O(\Gamma_k(G) + n^{1+\frac{1}{k+\ell+1}})$ by Lemma 2 and the definition of ε.

The number of edges added in step 2(a)i is $O(n)$ per iteration, and thus $O(n \log^2 n)$ over all $3 \log n$ iterations of step 2 and all choices of i.

The number of edges added in step 2(a)ivB due to $u \in S_i$ is at most $|S_i| \cdot |V_\ell| \cdot \left(2^i n^{-k\varepsilon} + \ell + 1\right) = O(n2^{-i}) n^{1-\ell\varepsilon} (2^i n^{-k\varepsilon} + \ell + 1) = O(n^{2-(k+\ell)\varepsilon} + n^{2-\ell\varepsilon} 2^{-i})$.

Notice that paths are only added if $2^i \geq n^{k\varepsilon}$, and so this expression is bounded by $O(n^{2-(k+\ell)\varepsilon})$. Hence, over all $3\log n$ iterations of step 2 and all choices of i, this is bounded by $O(n^{2-(k+\ell)\varepsilon}\log^2 n)$.

The number of edges added in step 2(a)ivB due to $u \in S_{i,j}$ for a fixed j is at most $|S_{i,j}|\cdot|V_\ell|\cdot(4k2^j+\ell+k) = O(n^{1-k\varepsilon}2^{-j})\cdot n^{1-\ell\varepsilon}\cdot(4k2^j+\ell+k) = O(n^{2-(k+\ell)\varepsilon})$, and so as we range over all j it is bounded by $O(n^{2-(k+\ell)\varepsilon}\log n)$. Since step 2(a)ivB is invoked $O(\log^2 n)$ times, using the definition of ε we see that the total number of edges in H meets the claimed bound.

Theorem 6. InterClusterSpanner *can be implemented in* $O(n^2\log^2 n)$ *time.*

Proof. Step 1 can be implemented in $O(m)$ time by Lemma 2. Fix some iteration of step 2a. Then steps 2ai, 2aii, and 2aiii can be implemented in $O(n+m)$ time. Step 2aivA runs in time $O((|S_i| + |\cup_{\text{all } j} S_{i,j}|)n2^i) = O(n^2) + O(|S_{i,q}|n2^i)$. Here we used that (1) $n^{1-k\varepsilon}2^{-j}$ is geometrically decreasing in j, (2) $|S_{i,j}| = \lceil n^{1-k\varepsilon}2^{-j}\rceil$, and (3) j is bounded above by $\lceil\log n^{1-k\varepsilon}\rceil$. Therefore, $|\cup_{\text{all } j} S_{i,j}| = O(|S_{i,q}|)$. Now, $|S_{i,q}| = O(n^{1-k\varepsilon}/(n^{-k\varepsilon}2^i)) = O(n/2^i)$. Hence, step 2a(iv)A runs in time $O(n^2)$. Over all iterations of step 2 and choices of i in step 2a, this gives $O(n^2\log^2 n)$ time.

By definition of FHEASW, step 2aivB runs in time $O(|V_\ell|(|S_i|2^i n^{-k\varepsilon} + \sum_j |S_{i,j}|(4k2^j))) = \tilde{O}(n^{1-\ell\varepsilon}(n2^{-i}\cdot 2^i n^{-k\varepsilon} + \sum_j n^{1-k\varepsilon}2^{-j}(4k2^j)))$, which equals $O(n^{2-(k+\ell)\varepsilon}\log n)$. Here we used the relation $2^i n^{-k\varepsilon} + \ell + 1 = \Theta(2^i n^{-k\varepsilon})$, which does not hold in general, but if we add the heavy edges along a path for $u \in S_i$ in step 2aivB we require that $2^i \geq n^{k\varepsilon}$, so in this case the relation holds. Over all invocations of step 2 and choices of i in step 2a, this results in $O(n^{2-(k+\ell)\varepsilon}\log^3 n)$ time, and thus is dominated by the time for iterations of step 2aivA.

Theorem 7. *With probability* $\geq 1-1/n$, H *is an additive* $(2k+4\ell)$*-spanner.*

Proof. Fix a pair $\{a,b\}$ of vertices in G and a shortest path P between them. We assume there is at least one heavy edge along P (recall that we have modified the definition of heavy), otherwise P will be added to F in step 1. Let $w_1,\ldots,w_r$ be the ordered sequence of vertices in P (w_1 is closest to a, and w_r is closest to b) that appear in both $\mathcal{C}_\ell$ and $\mathcal{C}_k$ (note that $r \geq 2$ since there is a heavy edge along P). Let $v_1,\ldots,v_r$ be the centers of the clusters in $\mathcal{C}_\ell$ containing $w_1,\ldots,w_r$, respectively.

Consider one iteration of step 2. We show that with probability at least $3/8$, using only the edges added to F in step 1 and the current iteration of step 2, there is a path of length at most $\delta_G(a,b)+2k+4\ell$ from a to b. Hence the probability that there is some iteration for which there is a path of length at most $\delta_G(a,b)+2k+4\ell$ is at least $1-(3/8)^{3\log n} \geq 1-1/n^3$. By a union bound, it will follow that for every pair of vertices, there is such a path with probability at least $1-1/n$, that is, H is an additive $(2k+4\ell)$-spanner.

Recall that an edge $\{u,v\}$ is of type i if $\min(\deg(u),\deg(v)) \in [2^i, 2^{i+1})$. We let i^* be such that all edges along P have type at most i^*, and there is at least one edge e^* of type i^*. We shall only consider the i^*-th iteration of step 2a. Notice that the entire path P is included in the edgeset E'.

Case 1: P contains at most $n^{-k\varepsilon}2^{i^*}$ heavy edges. Since e^* is of type i^*, one of its endpoints y has degree in the range $[2^{i^*}, 2^{i^*+1})$. Hence, all edges incident to y have type at most i^*, and are included in the graph (V, E'). Since S_{i^*} is a randomly chosen set of size $2n/2^{i^*}$, with probability at least $1-\left(1-\frac{2^*}{n}\right)^{2n/2^{i^*}} \geq 3/8$, there is a vertex $u \in S_{i^*} \cap N(y)$. We condition on this event.

Consider the walk Q from u to v_1 which first traverses edge $\{u, y\}$, then agrees with path P from y to w_1, then traverses at most ℓ edges along the spanning tree from w_1 to v_1 in the cluster centered at v_1. Observe that the number of heavy edges along this walk is at most $n^{-k\varepsilon}2^{i^*} + \ell + 1$. Also, since P contains at most $n^{-k\varepsilon}2^{i^*}$ heavy edges and at least 1 heavy edge, $2^{i^*} \geq n^{-k\varepsilon}$, as needed by step 2aivB. Moreover, by the triangle inequality, the walk Q is of length at most $\delta_G(w_1, y) + \ell + 1 \leq \delta_G(u, y) + 2\ell + 2 \leq \delta_G(u, y) + 2\ell + 2k$. It follows that in step 2aivB, there will be a walk Q' added to F from v_1 to u of length at most $\delta_{(V,E')}(w_1, y) + \ell + 1$. Similarly, there will be a walk Q'' added to F from u to v_r of length at most $\delta_{(V,E')}(y, w_r) + \ell + 1$. Hence, the walk from a to b which agrees with P from a until w_1, then goes to v_1, then takes the walk Q' to u, then the walk Q'' from u to v_r, then goes to w_r, then agrees with P from w_r to b, is of length at most $\delta_G(a, b) + 2 + 4\ell \leq \delta_G(a, b) + 2k + 4\ell$.

Case 2: P contains more than $n^{-k\varepsilon}2^{i^*}$ heavy edges. Let h denote the number of heavy edges along P. Since P is a shortest path, all vertices along P are distinct, and so there are at least h of them that appear in a cluster in $\mathcal{C}_k$. We need the slightly stronger property that there is a set T of $t \geq \lceil h/(2k) \rceil$ distinct vertices $v \in V_k$ for which some vertex in the cluster centered at v appears along P. Indeed, otherwise there would exist two vertices along P in the same cluster in $\mathcal{C}_k$ at a distance larger than $2k$ from each other, contradicting that P is a shortest path. Observe that $t \leq h + 1$, the number of heavy edges. Let $j^* \geq 0$ be such that $t \in [2^{j^*}, 2^{j^*+1})$. Since 2^j ranges from $n^{-k\varepsilon}2^{i^*}/(2k)$ to at least $n^{1-k\varepsilon}$, we can consider $j = j^*$ in step 2(a)ii.

Consider the following event $\mathcal{E} : S_{i^*,j^*} \cap T \neq \emptyset$. Then, $\Pr[\mathcal{E}]$ is at least $1 - \left(1 - \frac{t}{n^{1-k\varepsilon}}\right)^{|S_{i^*,j^*}|} \geq 1 - \left(1 - \frac{2^{j^*}}{n^{1-k\varepsilon}}\right)^{\lceil n^{1-k\varepsilon}2^{-j^*} \rceil} > \frac{3}{8}$. Conditioned on $\mathcal{E}$, let $v \in S_{i^*,j^*} \cap T$, and let u be the vertex in the cluster centered at v in $\mathcal{C}_k$ that lies along P. Consider the walk Q which first traverses the at most k edges on the spanning tree from v to u, then agrees with path P from u to w_1, then traverses the at most ℓ edges along the spanning tree from w_1 to v_1 in the cluster centered at v_1. The number of heavy edges along this walk is at most $h + \ell + k \leq 2kt + \ell + k \leq 4k2^{j^*} + \ell + k$. By the triangle inequality, the walk Q is of length at most $\delta_G(u, w_1) + k + \ell \leq \delta_G(v, v_1) + 2k + 2\ell$. It follows that in step 4aiiC, there will be a walk Q' added to F from v_1 to v of length at most $\delta_{(V,E')}(u, w_1) + \ell + k$. Similarly, there will be a walk Q'' added to F from v to v_r of length at most $\delta_{(V,E')}(u, w_r) + \ell + k$. Hence, the walk from a to b which agrees with P from a until w_1, then goes to v_1, then takes the walk Q' to v, then the walk Q'' from v to v_r, then goes to w_r, then agrees with P until b will have length at most $\delta_G(a, b) + \ell + 2(\ell + k) + \ell = \delta_G(a, b) + 2k + 4\ell$, as needed.

Acknowledgment. The author would like to thank Michael Elkin, Vitaly Feldman, Mihai Pătraşcu, Seth Pettie, and Anastasios Sidiropoulos.

References

1. Abraham, I., Gavoille, C., Malkhi, D.: On space-stretch trade-offs: upper bounds. In: SPAA, pp. 217–224 (2006)
2. Aingworth, D., Chekuri, C., Indyk, P., Motwani, R.: Fast estimation of diameter and shortest paths (without matrix multiplication). SIAM J. Comput. 28(4), 1167–1181 (1999)
3. Baswana, S., Kavitha, T., Mehlhorn, K., Pettie, S.: Additive spanners and (α, β)-spanners. ACM Transactions on Algorithms (2009)
4. Baswana, S., Kavitha, T.: Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In: FOCS, pp. 591–602 (2006)
5. Baswana, S., Kavitha, T., Mehlhorn, K., Pettie, S.: New constructions of (alpha, beta)-spanners and purely additive spanners. In: SODA, pp. 672–681 (2005)
6. Baswana, S., Sen, S.: Approximate distance oracles for unweighted graphs in $\tilde{o}(n^2)$ time. In: SODA, pp. 271–280 (2004)
7. Cohen, E.: Fast algorithms for constructing t-spanners and paths with stretch. SIAM J. Comput. 28(1), 210–236 (1998)
8. Cohen, E.: Polylog-time and near-linear work approximation scheme for undirected shortest paths. J. ACM 47(1), 132–166 (2000)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. The MIT Press & McGraw-Hill Book Company (2001)
10. Cowen, L.: Compact routing with minimum stretch. J. Algorithms 38(1), 170–183 (2001)
11. Cowen, L., Wagner, C.G.: Compact roundtrip routing in directed networks. J. Algorithms 50(1), 79–95 (2004)
12. Dor, D., Halperin, S., Zwick, U.: All-pairs almost shortest paths. SIAM J. Comput. 29(5), 1740–1759 (2000)
13. Elkin, M.: Personal communication (2009)
14. Elkin, M.: Computing almost shortest paths. ACM Transactions on Algorithms 1(2), 283–323 (2005)
15. Elkin, M., Peleg, D.: (1+epsilon, beta)-spanner constructions for general graphs. SIAM J. Comput. 33(3), 608–631 (2004)
16. Peleg, D., Ullman, J.D.: An optimal synchronizer for the hypercube. SIAM J. Comput. 18(4), 740–747 (1989)
17. Peleg, D., Upfal, E.: A trade-off between space and efficiency for routing tables. J. ACM 36(3), 510–530 (1989)
18. Pettie, S.: Low distortion spanners. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 78–89. Springer, Heidelberg (2007)
19. Roditty, L., Thorup, M., Zwick, U.: Deterministic constructions of approximate distance oracles and spanners. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 261–272. Springer, Heidelberg (2005)
20. Thorup, M., Zwick, U.: Compact routing schemes. In: SPAA (2001)
21. Thorup, M., Zwick, U.: Approximate distance oracles. J. ACM 52(1), 1–24 (2005)
22. Thorup, M., Zwick, U.: Spanners and emulators with sublinear distance errors. In: SODA, pp. 802–809 (2006)
23. Woodruff, D.P.: Lower bounds for additive spanners, emulators, and more. In: FOCS, pp. 389–398 (2006)

Composition Theorems in Communication Complexity

Troy Lee[1] and Shengyu Zhang[2]

[1] Rutgers University
troyjlee@gmail.com
[2] The Chinese University of Hong Kong
syzhang@cse.cuhk.edu.hk

Abstract. A well-studied class of functions in communication complexity are composed functions of the form $(f \circ g^n)(x,y) = f(g(x^1,y^1),\ldots,g(x^n,y^n))$. This is a rich family of functions which encompasses many of the important examples in the literature. It is thus of great interest to understand what properties of f and g affect the communication complexity of $(f \circ g^n)$, and in what way.

Recently, Sherstov [She09] and independently Shi-Zhu [SZ09b] developed conditions on the inner function g which imply that the quantum communication complexity of $f \circ g^n$ is at least the approximate polynomial degree of f. We generalize both of these frameworks. We show that the pattern matrix framework of Sherstov works whenever the inner function g is *strongly balanced*—we say that $g : X \times Y \to \{-1,+1\}$ is strongly balanced if all rows and columns in the matrix $M_g = [g(x,y)]_{x,y}$ sum to zero. This result strictly generalizes the pattern matrix framework of Sherstov [She09], which has been a very useful idea in a variety of settings [She08b, RS08, Cha07, LS09a, CA08, BHN09].

Shi-Zhu require that the inner function g has small *spectral discrepancy*, a somewhat awkward condition to verify. We relax this to the usual notion of discrepancy.

We also enhance the framework of composed functions studied so far by considering functions $F(x,y) = f(g(x,y))$, where the range of g is a group G. When G is Abelian, the analogue of the strongly balanced condition becomes a simple group invariance property of g. We are able to formulate a general lower bound on F whenever g satisfies this property.

1 Introduction

Communication complexity studies the minimum amount of communication needed to compute a function whose input variables are distributed between two or more parties. Since the introduction by Yao [Yao79] of an elegant mathematical model to study this question, communication complexity has grown into a rich field both because of its inherent mathematical interest and also its application to many other models of computation. See the textbook of Kushilevitz and Nisan [KN97] for a comprehensive introduction to the field.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 475–489, 2010.

In analogy with traditional computational complexity classes, one can consider different models of communication complexity based on the resources available to the parties. Besides the standard deterministic model, of greatest interest to us will be a randomized version of communication complexity, where the parties have access to a source of randomness and are allowed to err with some small constant probability, and a quantum model where the parties share a quantum channel and the cost is measured in qubits.

Several major open questions in communication complexity ask about how different complexity measures relate to each other. The log rank conjecture, formulated by Lovász and Saks [LS88], asks if the deterministic communication complexity of a Boolean function $F : X \times Y \to \{0,1\}$ is upper bounded by a polynomial in the logarithm of the rank of the matrix $[F(x,y)]_{x,y}$. Another major open question is if randomized and quantum communication complexity are polynomially related for all total functions. We should mention here that the assumption of the function being total is crucial as an exponential separation is known for a partial function [Raz99].

One approach to these questions has been to study them for restricted classes of functions. Many functions of interest are *block composed* functions. For finite sets X, Y, and E, a function $f : E^n \to \{-1,+1\}$, and a function $g : X \times Y \to E$, the block composition of f and g is the function $f \circ g^n : X^n \times Y^n \to \{-1,+1\}$ defined by $(f \circ g^n)(x,y) = f(g(x^1,y^1),\ldots,g(x^n,y^n))$ where $(x^i,y^i) \in X \times Y$ for all $i = 1,\ldots,n$. For example, if $E = \{-1,+1\}$, the inner product function results when f is PARITY and g is AND, set-intersection when f is OR and g is AND, and the equality function when f is AND and g is the function IS-EQUAL, which is one if and only if $x = y$.

In a seminal paper, Razborov [Raz03] gave tight bounds for the bounded-error quantum communication complexity of block composed functions where the outer function is symmetric and the inner function is bitwise AND. In particular, this result showed that randomized and quantum communication complexity are polynomially related for such functions.

More recently, very nice frameworks have been developed by Sherstov [She07, She09] and independently by Shi and Zhu [SZ09b] to bound the quantum complexity of block composed functions that goes beyond the case of symmetric f to work for any f provided the inner function g satisfies certain technical conditions. When g satisfies these conditions, these frameworks allow one to lower bound the quantum communication complexity of $f \circ g^n$ in terms of $\deg_\epsilon(f)$, the approximate polynomial degree of f, a well-studied measure.

Shi and Zhu are able to get a bound on $f \circ g^n$ in terms of the approximate degree of f whenever g is sufficiently "hard"—unfortunately, the hardness condition they need is in terms of "spectral discrepancy," a quantity which is somewhat difficult to bound, and their bound requires that g is a function on at least $\Omega(\log(n/d))$ bits, where d is the approximate polynomial degree of f. Because of this, Shi-Zhu are only able to reproduce Razborov's results with a polynomially weaker bound.

Sherstov developed so-called *pattern matrices* which are the matrix representation of a block composed function when g is a fixed function of a particularly nice form. Namely, in a pattern matrix the inner function $g : \{-1,+1\}^k \times ([k] \times \{-1,+1\}) \to \{-1,+1\}$ is parameterized by a positive integer k and defined by $g(x,(i,b)) = x_i \cdot b$, where x_i denotes the i^{th} bit of x. With this g, Sherstov shows that $\deg_\epsilon(f)$ is a lower bound on the quantum communication complexity of $f \circ g^n$, for any function f. Though seemingly quite special, pattern matrices have proven to be an extremely useful concept. First, they give a simple proof of Razborov's tight lower bounds for $f(x \wedge y)$ for symmetric f. Second, they have also found many other applications in unbounded-error communication complexity [She08b, RS08] and have been successfully extended to multiparty communication complexity [Cha07, LS09a, CA08, BHN09].

A key step in both the works of Sherstov and Shi-Zhu is to bound the spectral norm of a sum of matrices $\|\sum_i B_i\|$. This is the major step where these works differ. Shi-Zhu apply the triangle inequality to bound this as $\|\sum_i B_i\| \leq \sum_i \|B_i\|$. On the other hand, Sherstov observes that, in the case of pattern matrices, the terms of this sum are mutually orthogonal, *i.e.* $B_i^\dagger B_j = B_i B_j^\dagger = 0$ for all $i \neq j$. In this case, one has a stronger bound on the spectral norm $\|\sum_i B_i\| = \max_i \|B_i\|$.

We extend both the frameworks of Sherstov and Shi-Zhu. In the case of Shi-Zhu, we are able to reprove their theorem with the usual notion of discrepancy instead of the somewhat awkward spectral discrepancy. The main observation we make is that as all Shi-Zhu use in this step is the triangle inequality, we can repeat the argument with any norm here, including discrepancy itself.

In the case of pattern matrices, special properties of the spectral norm are used, namely the fact about the spectral norm of a sum of orthogonal matrices. We step back to see what key features of a pattern matrix lead to this orthogonality property. We begin with the Boolean case, that is, where the intermediate set E is taken to be $\{-1,+1\}$. In this case, a crucial concept is the notion of a *strongly balanced* function. We say that $g : X \times Y \to \{-1,+1\}$ is strongly balanced if in the sign matrix $M_g[x,y] = g(x,y)$ all rows and all columns sum to zero. We show that whenever the inner function g is strongly balanced, the key orthogonality condition holds; this implies that whenever g is strongly balanced and has discrepancy under the uniform distribution bounded away from one, the approximate degree of the outer function f is a lower bound on the quantum communication complexity of $f \circ g^n$.

We also consider the general case where the intermediate set is any group G. That is, we consider functions $F(x,y) = f(g(x,y))$, where $g : X \times Y \to G$ for a group G and $f : G \to \{-1,+1\}$ is a class function on G. The case $E = \{-1,+1\}$ discussed above corresponds to taking the group $G = \mathbb{Z}_2^n$. When G is a general Abelian group, the key orthogonality condition requires more than that the matrix $M_g[x,y] = g(x,y)$ is strongly balanced; still, it admits a nice characterization in terms of group invariance. A multiset $T \in G \times G$ is said to be G-invariant if $(s,s)T = T$ for all $s \in G$. The orthogonality condition will hold if and only if all pairs of rows and all pairs of columns of M_g (when viewed as multisets) are G-invariant. One can generalize the results discussed above to this general setting

with appropriate modifications. In the case that $G = \mathbb{Z}_2^n$, the G-invariant condition degenerates to the strongly balanced requirement of M_g.

2 Preliminaries

All logarithms are base two. For a complex number $z = a + ib$ we let $\bar{z} = a - ib$ denote the complex conjugate of z and $|z| = \sqrt{a^2 + b^2}$ and $\mathrm{Re}(z) = a$.

2.1 Complexity Measures

We will make use of several complexity measures of functions and matrices. Let $f : \{-1,+1\}^n \to \{-1,+1\}$ be a function. For $T \subseteq \{0,1\}^n$, the Fourier coefficient of f corresponding to the character χ_T is $\hat{f}_T = \frac{1}{2^n}\sum_x f(x)\chi_T(x) = \frac{1}{2^n}\sum_x f(x)\prod_{i\in T} x_i$. The *degree* of f as a polynomial, denoted $\deg(f)$, is the size of a largest set T for which $\hat{f}_T \neq 0$.

We reserve J for the all ones matrix, whose size will be determined by the context. For a matrix A let $A^\dagger$ denote the conjugate transpose of A. We use $A * B$ for the entrywise product of A, B, and $A \otimes B$ for the tensor product. If A is an m-by-n matrix then we say that $\mathrm{size}(A) = mn$. We use $\langle A, B\rangle = \mathrm{Tr}(AB^\dagger)$ for the inner product of A and B.

Let $\|A\|_1$ be the ℓ_1 norm of A, i.e. sum of the absolute values of entries of A, and $\|A\|_\infty$ the maximum absolute value of an entry. For a positive semidefinite matrix M let $\lambda_1(M) \geq \cdots \geq \lambda_n(M) \geq 0$ be the eigenvalues of M. We define the i^{th} singular value of A, denoted $\sigma_i(A)$, as $\sigma_i(A) = \sqrt{\lambda_i(AA^\dagger)}$. The rank of A, denoted $\mathrm{rk}(A)$ is the number of nonzero singular values of A. We will use several matrix norms. The spectral or operator norm is the largest singular value $\|A\| = \sigma_1(A)$, the trace norm is the summation of all singular values $\|A\|_{tr} = \sum_i \sigma_i(A)$, and the Frobenius norm is the ℓ_2 norm of the singular values $\|A\|_F = \sqrt{\sum_i \sigma_i(A)^2}$. When $AB^\dagger = A^\dagger B = 0$ we will say that A, B are *orthogonal*. Please note the difference with the common use of this term, which usually means $\langle A, B\rangle = 0$.

Fact 1. *For two matrices A, B of the same dimensions, if $AB^\dagger = A^\dagger B = 0$, then*

$$\mathrm{rk}(A+B) = \mathrm{rk}(A) + \mathrm{rk}(B),\ \ \|A+B\|_{tr} = \|A\|_{tr} + \|B\|_{tr},\ \ \|A+B\| = \max\{\|A\|, \|B\|\}.$$

Another norm we will use is the γ_2 norm, introduced to complexity theory in [LMSS07], and familiar in matrix analysis as the Schur product operator norm. The γ_2 norm can be viewed as a weighted version of the trace norm.

Definition 1

$$\gamma_2(A) = \max_{u,v:\|u\|=\|v\|=1} \|A * uv^\dagger\|_{tr}.$$

It is clear from this definition that $\gamma_2(A) \geq \|A\|_{tr}/\sqrt{mn}$ for an m-by-n matrix A.

For a norm Φ, the dual norm Φ^* is defined as $\Phi^*(v) = \max_{u:\Phi(u)\leq 1} |\langle u, v\rangle|$. The norm γ_2^*, dual to the γ_2 norm, looks as follows.

Definition 2

$$\gamma_2^*(A) = \max_{\substack{u_i, v_j: \\ \|u_i\|=\|v_j\|=1}} \sum_{i,j} A[i,j]\langle u_i, v_j\rangle.$$

Another complexity measure we will make use of is discrepancy.

Definition 3. *Let A be an m-by-n sign matrix and let P be a probability distribution on the entries of A. The discrepancy of A with respect to P, denoted* $\mathrm{disc}_P(A)$*, is defined as*

$$\mathrm{disc}_P(A) = \max_{x\in\{0,1\}^m, y\in\{0,1\}^n} |x^\dagger (A * P) y|.$$

We will write $\mathrm{disc}_U(A)$ for the special case where P is the uniform distribution. It is easy to see from this definition that $\mathrm{disc}_U(A) \leq \frac{\|A\|}{\sqrt{\mathrm{size}(A)}}$. Shaltiel [Sha03] has shown the deeper result that this bound is in fact polynomially tight:

Theorem 2 (Shaltiel). *Let A be a sign matrix. Then*

$$\frac{1}{108}\left(\frac{\|A\|}{\sqrt{\mathrm{size}(A)}}\right)^3 \leq \mathrm{disc}_U(A).$$

Discrepancy and the γ_2^* norm are very closely related. Linial and Shraibman [LS09c] observed that Grothendieck's inequality gives the following.

Theorem 3 (Linial-Shraibman). *For any sign matrix A and probability distribution P*

$$\mathrm{disc}_P(A) \leq \gamma_2^*(A * P) \leq K_G \,\mathrm{disc}_P(A)$$

where $1.67\ldots \leq K_G \leq 1.78\ldots$ is Grothendieck's constant.

Approximate measures. We will also use approximate versions of these complexity measures which come in handy when working with bounded-error models. Say that a function g gives an ϵ-approximation to f if $|f(x) - g(x)| \leq \epsilon$ for all $x \in \{-1,+1\}^n$. The ϵ-approximate polynomial degree of f, denoted $\deg_\epsilon(f)$, is the minimum degree of a function g which gives an ϵ-approximation to f. We will similarly look at the ϵ-approximate version of the trace and γ_2 norms. We give the general definition with respect to any norm.

Definition 4 (approximation norm). *Let $\Phi : \mathbb{R}^n \to \mathbb{R}$ be an arbitrary norm. Let $v \in \mathbb{R}^n$ be a sign vector. For $0 \leq \epsilon < 1$ we define the approximation norm Φ^ϵ as*

$$\Phi^\epsilon(v) = \min_{u:\|v-u\|_\infty \leq \epsilon} \Phi(u).$$

Notice that an approximation norm Φ^ϵ is not itself a norm—we have only defined it for sign vectors, and it will in general not satisfy the triangle inequality.

As a norm is a convex function, using the separating hyperplane theorem one can quite generally give the following equivalent dual formulation of an approximation norm.

Proposition 1. *Let $v \in \mathbb{R}^n$ be a sign vector, and $0 \leq \epsilon < 1$*

$$\Phi^{\epsilon}(v) = \max_u \frac{|\langle v, u\rangle| - \epsilon\|u\|_1}{\Phi^*(u)}$$

A proof of this can be found in the survey [LS09b].

2.2 Communication Complexity

Let X, Y, S be finite sets and $f : X \times Y \to S$ be a function. We will let $D(f)$ be the deterministic communication complexity of f, and $R_\epsilon(f)$ denote the randomized public coin complexity of f with error probability at most ϵ. We refer to the reader to [KN97] for a formal definition of these models. We will also study $Q_\epsilon(f)$ and $Q^*_\epsilon(f)$, the ϵ-error quantum communication complexity of f without and with shared entanglement, respectively. We refer the reader to [Raz03] for a nice description of these models.

For notational convenience, we will identify a function $f : X \times Y \to \{-1, +1\}$ with its sign matrix $M_f = [f(x, y)]_{x,y}$. Thus, for example, $\|f\|$ refers to the spectral norm of the sign matrix representation of f.

For all of our lower bound results we will actually lower bound the approximate trace norm or γ_2 norm of the function. Razborov showed that the approximate trace norm can be used to lower bound on quantum communication complexity, and Linial and Shraibman generalized this to the γ_2 norm.

Theorem 4 (Linial-Shraibman [LS09d]). *Let A be a sign matrix and $0 \leq \epsilon < 1/2$. Then*

$$Q^*_\epsilon(A) \geq \log\left(\gamma_2^{2\epsilon}(A)\right) - 2.$$

Composed functions. Before discussing lower bounds on a block composed function $f \circ g^n$, let us see what we expect the complexity of such a function to be. A fundamental idea going back to Nisan [Nis94] and Buhrman, Cleve, and Wigderson [BCW98], is that the complexity of $f \circ g^n$ can be related to the *query* complexity of f and the communication complexity of g. This holds true for deterministic, randomized, and quantum models of communication complexity and query complexity. For formal definitions of these measures and a survey of query complexity we recommend [BW02].

One advantage of working with block composed functions in light of this is that query complexity is in general better understood than communication complexity. In particular, a polynomial relationship between deterministic query complexity and degree, and randomized and quantum query complexities and approximate degree is known. Putting these two facts together gives the following corollary:

Corollary 1

$$D(f \circ g^n) = O(\deg(f)^4 D(g)), \quad R_{1/4}(f \circ g^n) = O(\deg_{1/4}(f)^6 R_{1/4}(g) \log \deg_{1/4}(f))$$

Our goal, then, in showing lower bounds on the complexity of a block composed function $f \circ g^n$ is to get something at least in the ballpark of this upper bound. Of course, this is not always possible. For example, when f is the PARITY function on n bits, and $g(x, y) = \oplus(x, y)$ this protocol just gives an upper bound of n bits, when the true complexity is constant. See recent results by Zhang [Zha09] and Sherstov [She10] for discussions on the tightness of the bounds in Corollary 1.

3 Rank of Block Composed Functions

We begin by analyzing the rank of a block composed function $f \circ g^n$ when the inner function g is strongly balanced. This case will illustrate the use of the strongly balanced assumption, and is simpler to understand than the bounded-error situation treated in the next section.

Let us first formally state the definition of strongly balanced.

Definition 5 (strongly balanced). *Let A be a sign matrix, and J be the all ones matrix of the same dimensions as A. We say that A is* balanced *if* $\mathrm{Tr}(AJ^\dagger) = 0$. *We further say that A is* strongly balanced *if $AJ^\dagger = A^\dagger J = 0$. We will say that a two-variable Boolean function is balanced or strongly balanced if its sign matrix representation is.*

Theorem 5. *Let $f : \{-1, +1\}^n \to \{-1, +1\}$ be an arbitrary function, and let g be a strongly balanced function. Then*

$$\mathrm{rk}(M_{f \circ g^n}) = \sum_{T \subseteq [n],\ \hat{f}_T \neq 0} \mathrm{rk}(M_g)^{|T|}.$$

Proof. Let us write out the sign matrix for $\chi_T \circ g^n$ explicitly. If we let $M_g^0 = J$ be the all ones matrix and $M_g^1 = M_g$, then we can nicely write the sign matrix representing $\chi_T(g(x^1, y^1), \ldots, g(x^n, y^n))$ as $M_{\chi_T \circ g^n} = \bigotimes_i M_g^{T[i]}$ where $T[i] = 1$ if $i \in T$ and 0 otherwise.

We see that the condition on g implies $M_{\chi_T \circ g^n} M^\dagger_{\chi_S \circ g^n} = 0$ if $S \neq T$. Indeed,

$$M_{\chi_T \circ g^n} M^\dagger_{\chi_S \circ g^n} = \left(\bigotimes_i M_g^{T[i]} \right) \left(\bigotimes_i M_g^{S[i]} \right)^\dagger = \bigotimes_i \left(M_g^{T[i]} (M_g^{S[i]})^\dagger \right) = 0.$$

This follows since, by the assumption $S \neq T$, there is some i for which $S[i] \neq T[i]$ which means that this term is either $M_g J^\dagger = 0$ or $J M_g^\dagger = 0$ because g is strongly balanced. The other case follows similarly.

Now that we have established this property, we can use Fact 1 to obtain

$$\mathrm{rk}(M_{f \circ g^n}) = \mathrm{rk}\Big(\sum_{T \subseteq [n]} \hat{f}_T \chi_T(g(x^1, y^1), \ldots, g(x^n, y^n)) \Big) = \sum_{\substack{T \subseteq [n] \\ \hat{f}_T \neq 0}} \mathrm{rk}(M_{\chi_T \circ g^n}) = \sum_{\substack{T \subseteq [n] \\ \hat{f}_T \neq 0}} \mathrm{rk}(M_g)^{|T|}$$

In the last step we used the fact that rank is multiplicative under tensor product.

In particular, this theorem means that whenever g is a strongly balanced function on a constant number of bits and $\mathrm{rk}(M_g) > 1$, then the log rank conjecture holds for $f \circ g^n$.

4 A Bound in Terms of Approximate Degree

In this section, we will address the frameworks of Sherstov and Shi-Zhu. We extend both of these frameworks to give more general conditions on the inner function g which still imply that the approximate degree of f is a lower bound on the quantum query complexity of the composed function $f \circ g^n$. In outline, both of these frameworks follow the same plan. By Theorem 4 it suffices to lower bound the approximate γ_2 norm (or approximate trace norm) of $f \circ g^n$. To do this, they use the dual formulation given by Proposition 1 and construct a witness matrix B which has non-negligible correlation with the target function and small γ_2^* (or spectral) norm.

A very nice way to construct this witness, used by both Sherstov and Shi-Zhu, is to use the *dual polynomial* of f. This is a polynomial v which certifies that the approximate polynomial degree of f is at least a certain value. More precisely, duality theory of linear programming gives the following lemma.

Lemma 1 (Sherstov [She09], Shi-Zhu [SZ09b]). *Let $f : \{-1,+1\}^n \to \{-1,+1\}$ and let $d = \deg_\epsilon(f)$. Then there exists a function $v : \{-1,+1\}^n \to \mathbb{R}$ such that*

1. *$\langle v, \chi_T \rangle = 0$ for every character χ_T with $|T| < d$.*
2. *$\langle v, f \rangle \geq \epsilon$.*
3. *$\|v\|_1 = 1$.*

Items (2),(3) are used to lower bound the correlation of the witness matrix with the target matrix and to upper bound the ℓ_1 norm of the witness matrix. In the most difficult step, and where these works diverge, Item (1) is used to upper bound the γ_2^* (or spectral) norm of the witness matrix. We treat each of these frameworks separately in the next two sections.

4.1 Sherstov's Framework

The proof of the next theorem follows the same steps as Sherstov's proof for pattern matrices (Theorem 5.1 [She09]). Our main contribution is to identify the strongly balanced condition as the key property of pattern matrices which enables the proof to work.

Theorem 6. *Let X, Y be finite sets, $g : X \times Y \to \{-1,+1\}$ be a strongly balanced function, and $M_g[x,y] = g(x,y)$ be the corresponding sign matrix. Let $f : \{-1,+1\}^n \to \{-1,+1\}$ be an arbitrary function. Then for any $\epsilon > 0$ and $\epsilon_0 > 2\epsilon$, we have*

$$Q_\epsilon^*(f \circ g^n) \geq \deg_{\epsilon_0}(f) \log_2 \left(\frac{\sqrt{|X||Y|}}{\|M_g\|} \right) - O(1).$$

Proof (Sketch). Let $d = \deg_{\epsilon_0}(f)$ and let v be a dual polynomial for f with properties as in Lemma 1. We define a witness matrix as

$$B[x,y] = \frac{2^n}{\text{size}(M_g)^n} v(g(x^1,y^1),\ldots,g(x^n,y^n))$$

We will use the matrix B to witness that $\|M_{f\circ g^n}\|_{tr}^{\epsilon_0}$ is large via Proposition 1. The theorem will then follow from Theorem 4.

There are three quantities to evaluate. One can compute $\langle M_{f\circ g^n}, B\rangle$ and $\|B\|_1$ using properties 2 and 3 of Lemma 1 respectively, together with the fact that as M_g is strongly balanced, it is in particular balanced.

The more interesting step is to bound $\|B\|$. As shown above, the strongly balanced property of g implies that the matrices $\chi_T \circ g^n$ and $\chi_S \circ g^n$ are orthogonal for distinct sets S, T and so Fact 1 can be used to greatly simplify this computation. Details are given in the full version.

Using the theorem of Shaltiel relating discrepancy to the spectral norm (Theorem 2), we get the following corollary:

Corollary 2. *Let the quantities be defined as in Theorem 6.*

$$Q^*_{1/8}(f\circ g^n) \geq \frac{1}{3}\deg_{1/3}(f)\big(\log\big(\frac{1}{\text{disc}_U(M_g)}\big) - 7\big) - O(1).$$

Comparison to Sherstov's pattern matrix: As mentioned in [She09], Sherstov's pattern matrix method can prove a quantum lower bound of $\Omega(\deg_\epsilon(f))$ for block composed functions $f\circ g^n$ if the matrix M_g contains the following 4×4 matrix S_4 as a submatrix:

$$S_4 = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix}, \qquad S_6 = \begin{bmatrix} 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 \\ 1 & -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 \end{bmatrix}$$

In this paper we show that the same lower bound holds as long as M_g contains a strongly balanced submatrix with discrepancy bounded away from one. Are there strongly balanced matrices not containing S_4 as a submatrix? It turns out that the answer is yes: we give the above 6×6 matrix S_6 as one example.

4.2 Shi-Zhu Framework

The method of Shi-Zhu does not restrict the form of the inner function g, but rather works for any g which is sufficiently "hard." The hardness condition they require is phrased in terms of a somewhat awkward measure they term spectral discrepancy.

Chattopadhyay [Cha08] extended the technique of Shi-Zhu to the case of multiparty communication complexity, answering an open question of Sherstov

[She08a]. In doing so, he gave a more natural condition on the hardness of g in terms of an upper bound on discrepancy frequently used in the multiparty setting and originally due to Babai, Nisan, and Szegedy [BNS92]. As all that is crucially needed is subadditivity, we do the argument here with γ_2^*, which is essentially equal to the discrepancy.

Theorem 7. *Let $f : \{-1,+1\}^n \to \{-1,+1\}$, and $g : X \times Y \to \{-1,+1\}$. Fix $0 < \epsilon < 1/2$, and let $\epsilon_0 > 2\epsilon$. Then*

$$Q_\epsilon^*(f \circ g^n) \geq \deg_{\epsilon_0}(f) - O(1).$$

provided there is a distribution μ which is balanced with respect to g and for which $\gamma_2^*(M_g * \mu) \leq \frac{\deg_{\epsilon_0}(f)}{2en}$.

Proof. We again use Proposition 1, this time with the γ_2 norm instead of the trace norm. To prove a lower bound we choose a witness matrix B as follows

$$B[x,y] = 2^n \cdot v(g(x^1,y^1),\ldots,g(x^n,y^n)) \cdot \prod_{i=1}^{n} \mu(x^i,y^i).$$

where v witnesses that f has approximate degree at least $d = \deg_{\epsilon_0}(f)$. This definition is the same as in the previous section where μ was simply the uniform distribution. As argued before, we have $\langle M_{f\circ g^n}, B\rangle \geq \epsilon_0$ and $\|B\|_1 = 1$ because $M_g * \mu$ is balanced.

The more interesting step is to upper bound $\gamma_2^*(B)$. We again expand B in terms of the Fourier coefficients of v. As we do not have special knowledge of the function g, we simply bound the resulting sum by the triangle inequality. Details are given in the full version.

5 A General Framework for Functions Composed through a Group

In this section we begin the study of general function compositions through a group G. In this case the inner function $g : X \times Y \to G$ has a group G as its range, and the outer function $f : G \to \{-1,+1\}$ is a class function, i.e. one that is invariant on conjugacy classes. Previous sections deal with the special case that $G = \mathbb{Z}_2^n$.

Let us recall the basic idea of the proof of Theorem 6. Following the work of Sherstov and Shi-Zhu [She09, SZ09b], to prove a lower bound on the quantum communication complexity for a composed function $f \circ g$, we constructed a witness matrix B which had non-negligible correlation with $f \circ g$ and small spectral norm. We used the dual polynomial p (of f) which has two important properties, first that p has non-negligible correlation with f and second that p has no support on low degree polynomials. We can then use the first property to show that the composed function $p \circ g$ will give non-negligible inner product with $f \circ g$ and the second to upper bound the spectral norm of $p \circ g$. The second

of these tasks is the more difficult. In the case of $G = \{-1, +1\}^n$, the degree of a character χ_T is a natural measure of how "hard" the character is — the larger T is, the smaller the spectral norm of $\chi_T \circ g$ will be. In the general group case, however, it is less clear what the corresponding "hard" and "easy" characters should be. In Section 5.1, we will show that this framework actually works for an arbitrary partition of the basis functions into Easy and Hard.

In carrying out this plan, one is still left with upper bounding $\|M_{p\circ g}\|$. Here, as in the Boolean case, it is again very convenient to have an orthogonality condition which can greatly simplify the computation of $\|M_{p\circ g}\|$ and give good bounds. In the Boolean case we have shown that M_g being strongly balanced implies this key orthogonality condition. In Section 5.2 and 5.3, we will show that for the general group, the condition is not only about each row and column of matrix M_g, but all pairs of rows and pairs of columns. In the Abelian group case, this reduces to a nice group invariance condition.

Even after applying the orthogonality condition to use the maximum bound instead of the triangle inequality for $\|M_{p\circ g}\|$, the remaining term $\|M_{\chi_i \circ g}\|$ (where χ_i is a "hard" character) is still not easy to upper bound. For block composed functions, fortunately, the tensor structure makes it feasible to compute. Section 5.4 gives a generalized version of Theorem 6.

5.1 General Framework

For a multiset T, $x \in T$ means x running over T. Thus $T = \{a(s) : s \in S\}$ means the multiset formed by collecting $a(s)$ with s running over S.

For a set S, denote by $L_{\mathbb{C}}(S)$ the $|S|$-dimensional vector space over the field $\mathbb{C}$ (of complex numbers) consisting of all linear functions from S to $\mathbb{C}$, endowed with inner product $\langle \psi, \phi \rangle = \frac{1}{|S|} \sum_{s \in S} \psi(s)\overline{\phi(s)}$. The distance of a function $f \in L_{\mathbb{C}}(S)$ to a subspace Φ of $L_{\mathbb{C}}(S)$, denoted by $d(f, \Phi)$, is defined as $\min\{\delta : \|f' - f\|_\infty \leq \delta, f' \in \Phi\}$, *i.e.* the magnitude of the least entrywise perturbation to turn f into Φ.

In the above setting, Theorem 6 generalizes to the following. The proof is along the line of that of Theorem 6 and is omitted in this conference version.

Theorem 8. *Consider a sign matrix $A = [f(g(x, y))]_{x,y}$ where $g : X \times Y \to S$ for a set S, and $f : S \to \{-1, +1\}$. Suppose that there are orthogonal basis functions $\Psi = \{\psi_i : i \in [|S|]\}$ for $L_{\mathbb{C}}(S)$. For any partition $\Psi = \Psi_{Hard} \uplus \Psi_{Easy}$, let $\delta = d(f, span(\Psi_{Easy}))$. If*

1. **(regularity)** *the multiset $\{g(x, y) : x \in X, y \in Y\}$ is a multiple of S, i.e. S repeated for some number of times.*
2. **(orthogonality)** *for all x, x', y, y' and all distinct $\psi_i, \psi_j \in \Psi_{Hard}$,*

$$\sum_y \psi_i(g(x, y))\overline{\psi_j(g(x', y))} = \sum_x \psi_i(g(x, y))\overline{\psi_j(g(x, y'))} = 0,$$

then

$$Q_\epsilon(A) \geq \log_2 \frac{\sqrt{MN} \cdot (\delta - 2\epsilon)}{\max_{\psi_i \in \Psi_{Hard}}(\max_g |\psi_i(g)| \cdot \|[\psi_i(g(x, y))]_{x,y}\|)} - O(1).$$

In the Boolean block composed function case, the regularity condition reduces to the matrix $[g(x,y)]$ being balanced, and later we will prove that the orthogonality condition reduces to the strongly balanced property.

5.2 Functions with Group Symmetry

For a general finite group G, two elements s and t are conjugate, denoted by $s \sim t$, if there exists an element $r \in G$ *s.t.* $rsr^{-1} = t$. Define H as the set of all class functions, *i.e.* functions f *s.t.* $f(s) = f(t)$ if $s \sim t$. Then H is an h-dimensional subspace of $L_{\mathbb{C}}(G)$, where h is the number of conjugacy classes. The irreducible characters $\{\chi_i : i \in [h]\}$ form an orthogonal basis of H. For a class function f and irreducible characters χ_i, let $\hat{f}_i = \langle \chi_i, f \rangle = \frac{1}{|G|} \sum_{g \in G} \chi_i(g)\overline{f(g)}$. An easy fact is that

$$|\hat{f}_i| = \frac{1}{|G|} \left| \sum_{g \in G} \chi_i(g)\overline{f(g)} \right| \leq \frac{1}{|G|} \sum_{g \in G} |f(g)||\chi_i(g)| \leq \left(\frac{1}{|G|} \sum_{g \in G} |f(g)| \right) \cdot \max_g |\chi_i(g)|. \tag{1}$$

In this section we consider the setting that S is a finite group G. In particular, we hope to have a better understanding of the orthogonality condition and the matrix operator norm $\|[\psi_(g(x,y))]_{x,y}\|$ in this setting.

The standard orthogonality of irreducible characters says that $\sum_{s \in G} \chi_i(s)\overline{\chi_j(s)} = 0$. The second condition in Theorem 8 is concerned with a more general case: For a multiset T with elements in $G \times G$, we need

$$\sum_{(s,t) \in T} \chi_i(s)\overline{\chi_j(t)} = 0, \qquad \forall i \neq j. \tag{2}$$

The standard orthogonality relation corresponds to the special that $T = \{(s,s) : s \in G\}$. We hope to have a characterization of a multiset T to make Eq. (2) hold.

We may think of the a multiset T with elements in set S as a function on S, with the value on $s \in S$ being the multiplicity of s in T. Since characters are class functions, for each pair (C_k, C_l) of conjugacy classes, only the value $\sum_{g_1 \in C_k, t \in C_l} T(g_1, t)$ matters for the sake of Eq. (2). We thus make T a class function by taking average within each class pair (C_k, C_l). That is, define a new function T' as

$$T'(s,t) = \sum_{s \in C_k, t \in C_l} T(s,t)/(|C_k||C_l|), \ \forall s \in C_k, \ \forall t \in C_l.$$

Proposition 2. *For a finite group G and a multiset T with elements in $G \times G$, the following three statements are equivalent:*

1. $\sum_{(s,t) \in T} \chi_i(s)\overline{\chi_j(t)} = 0, \ \forall i \neq j$
2. *T', as a function, is in span$\{\chi_i \otimes \overline{\chi_i} : i \in [h]\}$*
3. $[T'(s,t)]_{s,t} = C^\dagger D C$ *where D is a diagonal matrix and $C = [\chi_i(s)]_{i,s}$. That is, T', as a matrix, is normal and diagonalized exactly by the irreducible characters.*

5.3 Abelian Group

When G is Abelian, we have further properties to use. The first one is that $|\chi_i(g)| = 1$ for all i. The second one is that the irreducible characters are homomorphisms of G; that is, $\chi_i(st) = \chi_i(s)\chi_i(t)$. This gives a clean characterization of the orthogonality condition by group invariance. For a multiset T, denote by sT another multiset obtained by collecting all st where t runs over T. A multiset T with elements in $G \times G$ is *G-invariant* if it satisfies $(g, g)T = T$ for all $g \in G$. We can also call a function $T : G \times G \to \mathbb{C}$ G-invariant if $T(s, t) = T(rs, rt)$ for all $r, s, t \in G$. The overloading of the name is consistent when we view a multiset T as a function (counting the multiplicity of elements).

Proposition 3. *For a finite Abelian group G and a multiset T with elements in $G \times G$,*

$$T \text{ is } G\text{-invariant} \Leftrightarrow \sum_{(s,t)\in T} \chi_i(s)\overline{\chi_j(t)} = 0, \quad \forall i \neq j. \tag{3}$$

Another nice property of Abelian groups is that the orthogonality condition condition implies the regularity condition; see the full version for a proof. So what we finally get for Abelian groups is the following.

Corollary 3. *For a sign matrix $A = [f(g(x, y))]_{x,y}$ and an Abelian group G, if $d(f, span(Ch_{Easy})) = \Omega(1)$, and the multisets $S^{x,x'} = \{(g(x, y), g(x', y)) : y \in Y\}$ and $T^{y,y'} = \{(g(x, y), g(x, y')) : x \in X\}$ are G-invariant for any (x, x') and any (y, y'), then*

$$Q(A) \geq \log_2 \frac{\sqrt{MN}}{\max_{i\in Hard} \|[\chi_i(g(x, y))]_{x,y}\|} - O(1).$$

5.4 Block Composed Functions

We now consider a special class of functions g: block composed functions. Suppose the group G is a product group $G = G_1 \times \cdots \times G_t$, and $g(x, y) = (g_1(x^1, y^1), \cdots, g_t(x^t, y^t))$ where $x = (x^1, \cdots, x^t)$ and $y = (y^1, \cdots, y^t)$. That is, both x and y are decomposed into t components and the i-th coordinate of $g(x, y)$ only depends on the i-th components of x and y. The tensor structure makes all the computation easy. Theorem 6 can be generalized to the general product group case for arbitrary groups G_i.

Definition 6. *The ϵ-approximate degree of a class function f on product group $G_1 \times \cdots \times G_t$, denoted by $d_\epsilon(f)$, is the minimum d s.t. $\|f - f'\|_\infty \leq \epsilon$, where f' can be represented as a linear combination of irreducible characters with at most d non-identity component characters.*

Theorem 9. *For sign matrix $A = [f(g_1(x^1, y^1), \cdots, g_t(x^t, y^t)]_{x,y}$ where all g_i satisfy their orthogonality conditions, we have*

$$Q(A) \geq \min_{\{\chi_i\},S} \sum_{i\in S} \log_2 \frac{\sqrt{\text{size}(M_{g_i})}}{\deg(\chi_i)\|M_{\chi_i \circ g_i}\|} - O(1)$$

where the minimum is over all $S \subseteq [n]$ with $|S| > \deg_{1/3}(f)$, and all non-identity irreducible characters χ_i of G_i.

Previous sections, and [SZ09b], consider the case where all g_i's are the same and all G_i's are $\mathbb{Z}_2$. In this case, the above bound is equal to the one in Theorem 6, and the following proposition says that the group invariance condition degenerates to the strongly balanced property.

Proposition 4. *For $G = \mathbb{Z}_2^{\times t}$, the following two conditions for $g = (g_1, \cdots, g_t)$ are equivalent:*

1. *The multisets $S^{x,x'} = \{(g(x,y), g(x',y)) : y \in Y\}$ and $T^{y,y'} = \{(g(x,y), g(x,y')) : x \in X\}$ are G-invariant for any (x,x') and any (y,y'),*
2. *Each matrix $[g_i(x^i, y^i)]_{x^i, y^i}$ is strongly balanced.*

This equivalence does not in general hold if any group G_i has size larger than two.

Acknowledgments

We would like to thank Adi Shraibman for many enlightening conversations about these topics. TL is supported in part by a NSF postdoctoral research fellowship and by the grants CCF-0728937 and CCF-0832787. SZ is supported in part by Hong Kong grant RGC-419309.

References

[BBC+01] Beals, R., Buhrman, H., Cleve, R., Mosca, M., de Wolf, R.: Quantum lower bounds by polynomials. Journal of the ACM 48(4), 778–797 (2001); Earlier version in FOCS 1998

[BCW98] Buhrman, H., Cleve, R., Wigderson, A.: Quantum vs. classical communication and computation. In: Proceedings of the 30th ACM Symposium on the Theory of Computing, pp. 63–68 (1998)

[BHN09] Beame, P., Huynh-Ngoc, D.: Multiparty communication complexity and threshold circuit size of AC^0. In: Proceedings of the 50th IEEE Symposium on Foundations of Computer Science, pp. 53–62 (2009)

[BNS92] Babai, L., Nisan, N., Szegedy, M.: Multiparty protocols, pseudorandom generators for Logspace, and time-space trade-offs. Journal of Computer and System Sciences 45, 204–232 (1992)

[BW02] Buhrman, H., de Wolf, R.: Complexity measures and decision tree complexity: A survey. Theoretical Computer Science 288, 21–43 (2002)

[CA08] Chattopadhyay, A., Ada, A.: Multiparty communication complexity of disjointness. Technical Report TR-08-002, ECCC (2008)

[Cha07] Chattopadhyay, A.: Discrepancy and the power of bottom fan-in depth-three circuits. In: Proceedings of the 48th IEEE Symposium on Foundations of Computer Science, pp. 449–458 (2007)

[Cha08] Chattopadhyay, A.: Circuits, Communication, and Polynomials, PhD thesis, McGill University (2008)

[KN97] Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1997)

[LMSS07] Linial, N., Mendelson, S., Schechtman, G., Shraibman, A.: Complexity measures of sign matrices. Combinatorica 27(4), 439–463 (2007)
[LS88] Lovász, L., Saks, M.: Möbius functions and communication complexity. In: Proceedings of the 29th IEEE Symposium on Foundations of Computer Science, pp. 81–90 (1988)
[LS09a] Lee, T., Shraibman, A.: Disjointness is hard in the multiparty number-on-the-forehead model. Computational Complexity 18(2), 309–336 (2009)
[LS09b] Lee, T., Shraibman, A.: Lower bounds in communication complexity. Foundations and Trends in Theoretical Computer Science 3 (2009)
[LS09c] Linial, N., Shraibman, A.: Learning complexity versus communication complexity. Combinatorics, Probability, and Computing 18, 227–245 (2009)
[LS09d] Linial, N., Shraibman, A.: Lower bounds in communication complexity based on factorization norms. Random Structures and Algorithms 34, 368–394 (2009)
[LSŠ08] Lee, T., Shraibman, A., Špalek, R.: A direct product theorem for discrepancy. In: Proceedings of the 23rd IEEE Conference on Computational Complexity, pp. 71–80. IEEE, Los Alamitos (2008)
[Nis94] Nisan, N.: The communication complexity of threshold gates. In: Proceedings of Combinatorics, Paul Erdos is Eighty, pp. 301–315 (1994)
[NS94] Nisan, N., Szegedy, M.: On the degree of Boolean functions as real polynomials. Computational Complexity 4, 301–313 (1994)
[Raz99] Raz, R.: Exponential separation of quantum and classical communication complexity. In: Proceedings of the 31st ACM Symposium on the Theory of Computing, pp. 358–367 (1999)
[Raz03] Razborov, A.: Quantum communication complexity of symmetric predicates. Izvestiya: Mathematics 67(1), 145–159 (2003)
[RS08] Razborov, A., Sherstov, A.: The sign rank of AC^0. In: Proceedings of the 49th IEEE Symposium on Foundations of Computer Science, pp. 57–66 (2008)
[Sha03] Shaltiel, R.: Towards proving strong direct product theorems. Computational Complexity 12(1-2), 1–22 (2003)
[She07] Sherstov, A.: Separating AC^0 from depth-2 majority circuits. In: Proceedings of the 39th ACM Symposium on the Theory of Computing, pp. 294–301. ACM, New York (2007)
[She08a] Sherstov, A.: Communication lower bounds using dual polynomials. Bulletin of the EATCS 95, 59–93 (2008)
[She08b] Sherstov, A.: The unbounded-error communication complexity of symmetric functions. In: Proceedings of the 49th IEEE Symposium on Foundations of Computer Science (2008)
[She09] Sherstov, A.: The pattern matrix method. SIAM Journal on Computing (2009)
[She10] Sherstov, A.: On quantum-classical equivalence for composed communication problems. Quantum Information and Computation 10(5-6), 435–455 (2010)
[SZ09a] Shi, Y., Zhang, Z.: Communication complexities of XOR functions. Quantum information and computation 9(3-4), 255–263 (2009)
[SZ09b] Shi, Y., Zhu, Y.: Quantum communication complexity of block-composed functions. Quantum information and computation 9(5,6), 444–460 (2009)
[Yao79] Yao, A.: Some complexity questions related to distributive computing. In: Proceedings of the 11th ACM Symposium on the Theory of Computing, pp. 209–213 (1979)
[Zha09] Zhang, S.: On the tightness of the Buhrman-Cleve-Wigderson simulation. In: Proceedings of the 20th International Symposium on Algorithms and Computation, pp. 434–440 (2009)

Network Design via Core Detouring for Problems without a Core

Fabrizio Grandoni[1] and Thomas Rothvoß[2]

[1] Dipartimento di Informatica, Sistemi e Produzione,
Università di Roma Tor Vergata, Via del Politecnico 1, 00133 Roma, Italy
grandoni@disp.uniroma2.it
[2] Institute of Mathematics, EPFL, Lausanne, Switzerland
thomas.rothvoss@epfl.ch

Abstract. Some of the currently best-known approximation algorithms for network design are based on random sampling. One of the key steps of such algorithms is connecting a set of source nodes to a random subset of them. In a recent work [Eisenbrand,Grandoni,Rothvoß,Schäfer-SODA'08], a new technique, *core-detouring*, is described to bound the mentioned connection cost. This is achieved by defining a sub-optimal connection scheme, where paths are detoured through a proper connected subgraph (core). The cost of the detoured paths is bounded against the cost of the core and of the distances from the sources to the core. The analysis then boils down to proving the *existence* of a convenient core.

For some problems, such as connected facility location and single-sink rent-or-buy, the choice of the core is obvious (i.e., the Steiner tree in the optimum solution). Other, more complex network design problems do not exhibit any such core. In this paper we show that core-detouring can be nonetheless successfully applied. The basic idea is constructing a convenient core by manipulating the optimal solution in a proper (not necessarily trivial) way. We illustrate that by presenting improved approximation algorithms for two well-studied problems: virtual private network design and single-sink buy-at-bulk.

Keywords: Approximation algorithms, network design, virtual private network, buy-at-bulk, rent-or-buy, core detouring.

1 Introduction

In a seminal work, Gupta, Kumar, and Roughgarden [17] introduced a random-sampling-based framework to design and analyze approximation algorithms for network design. This way, they achieved improved approximation algorithms for three relevant network design problems: virtual private network design, single-sink rent-or-buy, and single-sink buy-at-bulk (see also [4,5,12,21]). Generalizations and adaptations of their approach were later successfully applied to several other problems, including multi-commodity rent-or buy [1,8,16], connected facility location [6], stochastic (online) Steiner tree [8,9,18], universal TSP [9,26] and many others.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 490–502, 2010.

One of the key ingredients in Gupta et al.'s approach [17] is connecting a set of source nodes to a randomly and independently sampled subset of them. The shortest-path distances from the source set to the sampled subset are then bounded against the cost of an optimum Steiner tree over the sampled nodes. In a recent work [6], Eisenbrand, Grandoni, Rothvoß, and Schäfer gave an improved analytical tool, *core detouring*, to bound the connection cost above. The crux of their method is designing a sub-optimal connection scheme, and bounding its cost. In their scheme connection paths are detoured through a proper connected subgraph (*core*). More formally, consider an undirected graph G with edge weights $\{c_e\}_{e\in E}$. We let $\ell(v,u)$ be the shortest path distance between v and u, and $\ell(v,U) := \min_{u\in U}\{\ell(v,u)\}$ for any $U \subseteq V(G)$. Let also $c(E') := \sum_{e\in E'} c_e$ for any $E' \subseteq E(G)$. To lighten the notation, we sometimes use G' instead of $E(G')$ or $V(G')$, where the meaning will be clear from the context.

Theorem 1 (Core Detouring) [6]. *Given an undirected graph $G = (V, E)$, with edge weights*[1] *$\{c_e\}_{e\in E}$, clients $C \subseteq V$, a connected subgraph G', a root $z \in V(G')$ and $p \in (0, 1]$. Mark each client independently with probability p, and denote the marked clients by C'. Then $E[\sum_{v\in C} \ell(v, C' \cup \{z\})] \leq (e^{-p|C|}|C| + \frac{0.8067}{p})c(G') + 2\sum_{v\in C} \ell(v, G')$.*

To have an intuition of the proof of the theorem, imagine G' as a cycle of length $|C|$, and think of the shortest paths $\ell(v, G')$ as edges $(v, f(v))$, where $f : C \to V(G')$ is a bijective function. This can be enforced by duplicating the edges of G', computing an Euler tour of the new graph, and performing node duplications and edge contractions in a proper way. Now connect each client $v \in C$ to the closest sampled client $v' \in C'$ in terms of number of hops (disregarding edge weights). It is not hard to see that each edge $(v, f(v))$ is used twice in expectation (once to approach G' and once to leave it). This accounts for the factor 2 in the upper bound. Moreover, each edge of G' is used by $\frac{1}{2p}$ connection paths in expectation, which becomes $\frac{1}{p}$ in the upper bound due to edge duplication. The refined factor $(|C|e^{-p|C|} + 0.8067/p)$ is obtained by flow canceling, and a more involving analysis. We remark that in the argument above the connectivity of G' is crucial.

Our Results and Techniques. The Core Detouring Theorem is existential in flavor: it is sufficient to show the existence of a convenient core G', of small cost and sufficiently close to the clients C. For some network design problems, a natural candidate is provided by the structure of the optimum solution. For example, the optimum solution for connected facility location and single-sink rent-or-buy contains a Steiner tree T. Applying the Core Detouring Theorem to T leads to improved approximation algorithms for those two problems [6].

In this paper we show that core-detouring can be successfully applied to other network design problems, where the optimum solution does not exhibit any convenient core. The basic idea here is constructing one such core by manipulating the optimal solution in a proper way. We illustrate that by presenting improved

[1] Throughout this paper we use the terms weight and cost interchangeably.

approximation algorithms for virtual private network design and single-sink buy-at-bulk. As we will see, the construction of a good core for the considered problems involves a few non-trivial ideas.

Virtual Private Network Design (VPN). VPN models scenarios where the traffic pattern is uncertain or rapidly changing, and henceforth the network must be able to support a family of traffic matrices. This family is implicitly expressed by upper bounding the amount of traffic which each node can send and receive.

Virtual Private Network Design (VPN). Given an undirected graph $G = (V, E)$, with edge weights $\{c_e\}_{e\in E}$, a set S of *senders* and R of *receivers*, with out-traffic bounds $\{b_s^+\}_{s\in S}$ and in-traffic bounds $\{b_r^-\}_{r\in R}$. A traffic matrix $M = \{M_{sr}\}_{(s,r)\in S\times R}$ is feasible if $\sum_{r\in R} M_{sr} \leq b_s^+$ and $\sum_{s\in S} M_{sr} \leq b_r^-$ for all $s \in S$ and $r \in R$. Find a minimum cost $\sum_{e\in E} c_e x_e$ capacity reservation $\{x_e\}_{e\in E}$ and paths $\{P_{sr}\}_{(s,r)\in S\times R}$ such that, for every feasible traffic matrix M, it is possible to route a flow M_{sr} along path P_{sr} simultaneously for all $(s, r) \in S \times R$, without exceeding the capacity x_e reserved on each edge e.

Following the literature on the problem, w.l.o.g. and unless differently stated, we assume $b_s^+ = b_r^- = 1$ for all $s \in S, r \in R$, and we let $|S| \leq |R|$. Possibly, $S \cap R \neq \emptyset$. We remark that a solution to VPN can be encoded by providing the paths P_{sr} only. In fact, for a given choice of the paths, the optimal choice for each x_e is the maximum cardinality of a matching in the bipartite graph $G_e = (S \cup R, E_e)$, where $sr \in E_e$ iff $e \in P_{sr}$. The current best approximation ratio for VPN is 3.39 [2,5].

Theorem 2. *There is an expected* 2.80*-approximation algorithm for VPN.*

The proof of Theorem 2 is based on the following two main ideas. We first show that the input VPN instance $\mathcal{I}$ is equivalent (in expectation) to a different VPN instance $\mathcal{I}_s$ with the same set of receivers, and a unique random sender s with out-traffic bound $|S|$. Here we crucially exploit König's theorem: *in a bipartite graph the maximum cardinality of a matching equals the minimum cardinality of a vertex cover*. In particular, König's theorem implies that graph G_e has a vertex cover C_e of cardinality $|C_e| = x_e$ for any given choice of the paths P_{sr}.

Consider the following problem:

Single-Sink Rent-or-Buy (SROB). Given an undirected graph $G = (V, E)$, with edge weights $\{c_e\}_{e\in E}$, a root z, a parameter $M \in \mathbb{Q}^+$ and clients $D \subseteq V$. Find a set of paths $\{P_{zv}\}_{v\in D}$ so as to minimize $\sum_{e\in E} c_e \min\{M, |\{P_{zv} \mid e \in P_{zv}\}|\}$.

We remark that an optimal solution to SROB consists of a Steiner tree containing the root and whose edges support at least M paths each, and a shortest path from each client to the Steiner tree. The second step of our proof is showing that, for any s, $\mathcal{I}_s$ is (deterministically) equivalent to an SROB instance $\mathcal{I}'_s$ with clients $D = R$, root $z = s$ and parameter $M = |S|$. We achieve the claimed approximation guarantee by applying the Core Detouring Theorem to the Steiner tree in $OPT_{\text{SROB}}(\mathcal{I}'_s)$[2].

[2] For a problem $\mathcal{P}$ and an instance $\mathcal{I}$ of $\mathcal{P}$, we let $OPT_{\mathcal{P}}(\mathcal{I})$ denote the optimal solution to $\mathcal{P}$ on $\mathcal{I}$. We use $OPT_{\mathcal{P}}(\mathcal{I})$ also to denote the cost of the optimal solution.

Single-Sink Buy-at-Bulk (SSBB). SSBB is another prototypical network design problem, which is used to model scenarios where the capacity is reserved on edges in a discrete fashion to support a given traffic matrix. (For a comparison, in the case of VPN the traffic is unknown but the capacity is installed in a continuous fashion). This is formalized by defining a set of cable types, each one characterized by a cost (per unit length) and a capacity. We are allowed to install $n_{i,e} \geq 0$ copies of each cable type i on edge e.

Single-Sink Buy-at-Bulk (SSBB). Given an undirected graph $G = (V, E)$, with edge weights $\{c_e\}_{e \in E}$, a set of source nodes D and a sink node r. Given a set of *cable types* $1, 2, \ldots, k$, with capacities $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_k$ and costs $\sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_k$. Assume $\delta_i := \frac{\sigma_i}{\mu_i}$ is a decreasing function of i (*economies of scale*). Find a cable installation $\{n_{i,e}\}_{1 \leq i \leq k, e \in E}$, with $n_{i,e} \in \mathbb{N}$, minimizing $\sum_{i,e} c_e \, \sigma_i \, n_{i,e}$ and such that one unit of flow can be routed simultaneously from each source node to the sink without exceeding the capacity $\sum_i \mu_i \, n_{i,e}$ on each edge e.

Depending on whether the flow originating at a given source can be routed along several paths or not, we distinguish between *splittable* SSBB (s-SSBB) and *unsplittable* SSBB (u-SSBB), respectively. The best-known approximation bounds for s-SSBB and u-SSBB are 23.93 [2,12] and 148.48 [2,21], respectively.

Theorem 3. *There is an expected* 20.41*-approximation algorithm for s-SSBB.*

The best-known approximation algorithms for s-SSBB [12,17,21] are based on random sampling. These algorithms consist of a sequence of aggregation rounds. In their analysis, in order to upper bound the cost of cables installed at round t, it is convenient to consider the cost paid by the optimum solution to install cables of type larger than s, for a proper s. That subset of cables induces a graph G_s which is in general disconnected. This rules out a direct application of the Core Detouring Theorem. For this reason, we developed the following generalized version of that theorem, which applies also to the case G' is disconnected. For a given subgraph G', we let $\ell_{G'}(v, w)$ be the distance from v to w in the graph resulting from the contraction (of the connected components) of G'.

Theorem 4 (Multi-Core Detouring). *Given an undirected graph $G = (V, E)$, with edge weights $\{c_e\}_{e \in E}$, clients $C \subseteq V$, a subgraph G', a root z and $p \in (0, 1]$. Mark each client independently with probability p, and denote the marked clients by C'. Then $E[\sum_{v \in C} \ell(v, C' \cup \{z\})] \leq (e^{-p|C|}|C| + \frac{0.8067}{p}) c(G') + 2 \sum_{v \in C} \ell_{G'}(v, z)$.*

The theorem above is achieved by embedding the shortest paths in the contracted graph into the original graph, and considering the graph G'' induced by edges crossed by a large enough number of paths. This graph is connected and contains the root. The Core Detouring Theorem is then applied to G''. With respect to our applications, we can think of Theorem 4 as a way of extracting from the optimum solution (providing G') a convenient core.

We describe a simple polynomial-time procedure, inspired by an existential proof by Karger and Minkoff [22], to transform any solution S to s-SSBB into

a tree solution[3] U on the same input instance, while increasing the cost of the solution at most by a factor 2.

Theorem 5. *For any given solution S to an s-SSBB instance, there is a polynomial-time procedure to construct a tree solution U for the same instance of cost at most twice the original cost.*

Being U feasible for the corresponding u-SSBB instance, we obtain the following corollary.

Corollary 1. *Given a ρ-approximation algorithm for s-SSBB, there is a 2ρ-approximation algorithm for u-SSBB. In particular, there is a $2 \cdot 20.41 = 40.82$ approximation algorithm for u-SSBB.*

The results concerning VPN and SSBB are described in Sections 2 and 3, respectively.

Related Work. VPN was independently defined by Fingerhut et al. [7], and by Duffield et al. [3] and since then, studied by various authors in several variations. The version that we refer to is also called *asymmetric* VPN. This problem is NP-hard even when we restrict to tree solutions [15]. Constant approximation algorithms are presented in [5,15,17,28]. It is known that the optimum solution is not always a tree. Curiously, the algorithms in [15,17] construct a tree solution, while the current best algorithm in [5] does not. We will use a variant of the latter algorithm to achieve our improved bound.

A 2-approximation is known [5] for the *balanced* case $|S| = |R|$, which improves on the 3-approximation in [20]. In [20] it is proved that an optimal tree solution for the balanced case can be computed in polynomial time. Very recently [24], it has been shown that the optimal solution is not a tree even in the balanced case, and that the problem remains NP-hard in that special case as well. In the same paper, a $(2+\varepsilon)$-approximation for the *almost balanced* case $|R|/|S| = O(1)$ was stated, for an arbitrary but fixed $\varepsilon > 0$.

In *symmetric* VPN the traffic is undirected, and each terminal v has a unique threshold b_v which upper bounds the amount of traffic which v is responsible for. In [15] a 2-approximation is given for symmetric VPN. In the same paper the authors show that an optimal tree solution can be computed in polynomial time. The so-called *VPN conjecture* states that symmetric VPN always has an optimal tree solution, and hence can be solved in polynomial time. In a breakthrough paper [11], this conjecture was recently proved to be true (see also [13,19] for former proofs of the conjecture on ring networks, which introduce part of the ideas used in [11]).

SSBB has been extensively studied in the literature. It is NP-hard, e.g., by reduction from the Steiner tree problem. Meyerson, Munagala, and Plotkin [23] gave an $O(\log n)$ approximation for s-SSBB. Garg, Khandekar, Konjevod, Ravi, Salman, and Sinha [10] described an $O(k)$ approximation, where k is the number of cable types. The first constant approximation is due to Guha, Meyerson,

[3] A tree solution is a solution where edges with non-zero capacity induce a tree.

(U1) Choose a receiver $r^* \in R$ uniformly at random.
(U2) Mark each receiver uniformly at random with probability $\frac{\alpha}{|S|}$. Term R' as the marked receivers.
(U3) For each $s \in S$, compute a ρ_{st}-approximative Steiner tree T_s spanning $\{s, r^*\} \cup R'$ and install cumulatively 1 unit of capacity on T_s.
(U4) Install 1 unit of capacity cumulatively on the shortest path from each receiver r to the closest node in $R' \cup \{r^*\}$.

Fig. 1. Algorithm `vpn` for VPN

and Munagala [14]: the approximation ratio of their algorithm is roughly 2000. This approximation was reduced to 216 by Talwar [27]. Gupta, Kumar, and Roughgarden [17] described an improved 76.8 approximation algorithm, based on random sampling. Refining their approach, the approximation was later reduced to 65.49 by Jothi and Raghavachari [21], and eventually to 24.92 by Grandoni and Italiano [12].

The unsplittable case u-SSBB is less studied, though probably more interesting from an application point of view. The algorithm by Talwar is a 216-approximation for u-SSBB as well. Unfortunately, this is not the case for the following improved random-sampling algorithms (i.e., those algorithms do not guarantee that the flow is unsplittable). Jothi and Raghavachari [21] show how to transform the 76.8 approximation algorithm for s-SSBB by Gupta et al. [17] into a $2 \cdot 76.8 = 153.6$ approximation algorithm for u-SSBB. Their approach is algorithm-specific: it would not work with an (even slightly) different algorithm. (In particular, it cannot be applied to our s-SSBB algorithm nor to the s-SSBB algorithms in [12,21]). In contrast, the reduction from Corollary 1 can use any s-SSBB algorithm as a black box.

SROB [15,22,25] is the special case of SSBB where there are only two cable types, one of very small capacity and cost per unit capacity $\delta_1 = 1$, and the other of cost $\sigma_2 = M \geq 1$ and very large capacity. The current best approximation ratio for SROB is 2.92 [6].

The recent result in [2], trivially implies improved approximation factors 3.39, 2.80, 23.93, and 148.48 for VPN, SROB, s-SSBB, and u-SSBB, respectively.

2 Virtual Private Network Design

In this section we present our improved 2.80-approximation algorithm for VPN, hence proving Theorem 2. Having in mind that for any fixed $\delta > 0$, there is a $(2 + \delta\frac{|R|}{|S|})$-approximation algorithm for VPN [24] (recall that $|R| \geq |S|$), we may assume that $|S| \leq \varepsilon|R|$ for an arbitrarily small $\varepsilon > 0$.

Algorithm `vpn`, which is a slight adaptation of the VPN algorithm in [5], is described in Figure 1. The quantity α is a positive constant to be fixed later. The best-known approximation factor for the Steiner tree problem is denoted by ρ_{st}. Currently $\rho_{st} < 1.39$ [2].

For a given VPN instance $\mathcal{I}$ and any sender $s \in S$, we let $\mathcal{I}_s$ be the VPN instance with the same receiver set as $\mathcal{I}$, and with sender set $\{s\}$, where the out-traffic bound for s is $|S|$. (Recall that $b_s^+ = 1$ for the original problem). The following lemma crucially exploits König's Theorem.

Lemma 1. $\sum_{s\in S} OPT_{\text{VPN}}(\mathcal{I}_s) \leq |S| \cdot OPT_{\text{VPN}}(\mathcal{I})$.

Proof. We show that, for a random sender s^*, $E[OPT_{\text{VPN}}(\mathcal{I}_{s^*})] \leq OPT_{\text{VPN}}(\mathcal{I})$. Let $\{P_{sr}\}_{(s,r)\in S\times R}$ be the optimal paths for $\mathcal{I}$ and let $\{x_e\}_{e\in E}$ be the induced capacities. Consider the solution to $\mathcal{I}_{s^*}$ induced by paths $\{P_{s^*r}\}_{r\in R}$ and let $\{x'_e\}_{e\in E}$ be the corresponding capacity reservation. Consider the bipartite graph $G_e = (S \cup R, E_e)$, with $sr \in E_e$ iff $e \in P_{sr}$. Let $C_e \subseteq S \cup R$ be a vertex cover for G_e of size x_e (which exists by König's theorem). Clearly $x'_e \leq \min\{|N_e(s^*)|, |S|\}$, whereby $N_e(s^*)$ are the nodes adjacent to s^* in G_e. Let us show that $E[x'_e] \leq x_e$. The event $\{s^* \in S \cap C_e\}$ happens with probability $\frac{|S\cap C_e|}{|S|}$. In this case we can upper bound x'_e with $|S|$. In the complementary case we can upper bound x'_e with $|N_e(s^*)| \leq |R \cap C_e|$, where we exploit the fact that s^* can be only adjacent to nodes of $R \cap C_e$ (otherwise C_e would not be a vertex cover). Altogether

$$E[x'_e] \leq \frac{|S \cap C_e|}{|S|} \cdot |S| + \left(1 - \frac{|S \cap C_e|}{|S|}\right) \cdot |R \cap C_e| \leq |S \cap C_e| + |R \cap C_e| = |C_e| = x_e.$$

The claim follows since

$$E[OPT_{\text{VPN}}(\mathcal{I}_{s^*})] \leq E\Big[\sum_{e\in E} c_e x'_e\Big] \leq \sum_{e\in E} c_e x_e = OPT_{\text{VPN}}(\mathcal{I}). \qquad \square$$

Let $\mathcal{I}'_s$ be the SROB instance with clients $D = R$, root $z = s$ and parameter $M = |S|$.

Lemma 2. $OPT_{\text{SROB}}(\mathcal{I}'_s) = OPT_{\text{VPN}}(\mathcal{I}_s)$.

Proof. Consider any set of paths $\{P_{sr}\}_{r\in R}$. It is sufficient to show that, for any given edge e, the corresponding cost associated to edge e is the same in the SROB and VPN case. The cost paid in the SROB case is by definition $c(e) \cdot \min\{M, |\{P_{sr} : e \in P_{sr}\}|\}$. With respect to VPN, consider the set of receivers $R_e := \{r \in R \mid e \in P_{sr}\}$ using edge e. In the worst case a traffic matrix routes $k' := \min\{|S|, |R_e|\}$ units of flow along e. Hence, also in this case the cost associated to e is $c(e) \cdot \min\{|S|, |R_e|\} = c(e) \cdot \min\{M, |\{P_{sr} : e \in P_{sr}\}|\}$. $\square$

Let C_s be the Steiner tree in $OPT_{\text{SROB}}(\mathcal{I}'_s)$, and $U_{s,r}$ the shortest path from $r \in R$ to C_s. Define $\Sigma_S := \sum_{s\in S} c(C_s)$ and $\Sigma_C := \frac{1}{M}\sum_{s\in S}\sum_{r\in R} c(U_{s,r})$. First we upper bound the cost of the $M = |S|$ Steiner trees computed by `vpn`.

Lemma 3. $E[\sum_{s\in S} c(T_s)] \leq \rho_{st} \cdot \Sigma_S + \rho_{st}(\alpha + \varepsilon) \cdot \Sigma_C$.

Proof. Recall that $M/|R| = |S|/|R| \leq \varepsilon$. For each s take the core C_s and attach the path $U_{s,r}$ for all $r \in R'$. Each $U_{s,r}$, $r \in R$, is used with probability at most $\frac{\alpha}{M} + \frac{1}{|R|}$, thus there exists a Steiner tree over $\{s, r^*\} \cup R'$ of expected cost

$c(C_s) + (\frac{\alpha}{M} + \frac{1}{|R|}) \sum_{r \in R} c(U_{s,r})$. Multiplying this quantity by the Steiner tree approximation factor, and summing over all s we obtain

$$\sum_{s \in S} E[c(T_s)] \leq \rho_{st}\Big(\sum_{s \in S} c(C_s) + \Big(\frac{\alpha}{M} + \frac{1}{|R|}\Big) \sum_{s \in S} \sum_{r \in R} c(U_{s,r})\Big) \leq \rho_{st} \Sigma_S + \rho_{st}(\alpha + \varepsilon)\Sigma_C . \square$$

We next bound the cost of connecting each receiver to the closest node in $R' \cup \{r^*\}$ via the Core Detouring Theorem.

Lemma 4. $E[\sum_{r \in R} \ell(r, R' \cup \{r^*\})] \leq \frac{0.81}{\alpha} \Sigma_S + 2\Sigma_C$.

Proof. Let $s \in S$. Applying the Core Detouring Theorem with $C = R$, $G' = C_s$, $z = r^*$ and $p = \alpha/M$, $E[\sum_{r \in R} \ell(r, R' \cup \{r^*\})] \leq (e^{-\frac{\alpha}{M}|R|}|R| + \frac{0.8067}{\alpha/M}) c(C_s) + 2 \sum_{r \in R} c(U_{s,r}) \leq \frac{0.81M}{\alpha} c(C_s) + 2 \sum_{r \in R} c(U_{s,r})$, where we use the assumption $|R| \gg |S| = M$. Averaging this bound over all s, we obtain

$$E\Big[\sum_{r \in R} \ell(r, R' \cup \{r^*\})\Big] \leq \frac{0.81M}{\alpha} \sum_{s \in S} \frac{1}{M} c(C_s) + 2 \frac{1}{M} \sum_{s \in S} \sum_{r \in R} c(U_{s,r}) = \frac{0.81}{\alpha} \Sigma_S + 2\Sigma_C . \square$$

Theorem 6. *For a suitable choice of α and $|R|/|S|$ large enough, Algorithm* `vpn` *gives an expected* 2.80 *approximation for VPN.*

Proof. From Lemmas 1 and 2, $\Sigma_S + \Sigma_C = 1/M \cdot \sum_{s \in S} \big(M\, c(C_s) + \sum_{r \in R} c(U_{s,r})\big)$ $= 1/M \cdot \sum_{s \in S} OPT_{\text{SROB}}(\mathcal{I}'_s) = 1/M \cdot \sum_{s \in S} OPT_{\text{VPN}}(\mathcal{I}_s) \leq OPT_{\text{VPN}}$. By Lemmas 3 and 4, the expected cost of the solution computed by the algorithm is

$$(\rho_{st} \cdot \Sigma_S + \rho_{st}(\alpha + \varepsilon)\Sigma_C) + \Big(\frac{0.81}{\alpha}\Sigma_S + 2\Sigma_C\Big) \overset{\alpha = 0.5748}{\leq} 2.80(\Sigma_C + \Sigma_S) \leq 2.80 \cdot OPT_{\text{VPN}} . \square$$

Theorem 2 then follows.

3 Single-Sink Buy-at-Bulk

In this section we present our improved algorithms for SSBB. We start with the proof of the Multi-Core Detouring Theorem. Then we present our results for the unsplittable and splittable case.

Multi-Core Detouring

Proof (of Theorem 4). Let $\phi_p(C, G') := 2 \sum_{v \in C} \ell_{G'}(v, z) + \gamma\, c(G')$ with $\gamma := (|C| \cdot e^{-p|C|} + \frac{0.8067}{p})$. We will find a connected subgraph G'' of G with $z \in V(G'')$, having $\phi_p(C, G'') \leq \phi_p(C, G')$. The claim then follows by applying the Core Detouring Theorem to G''. Let P_{vz} be the path, attaining the length $\ell_{G'}(v, z)$, i.e. it is a shortest v-z path in G, where edges in G' account with cost 0. Since these paths are shortest paths, we may assume that $\bigcup_{v \in C} P_{vz}$ induces a tree T, rooted at z. For $e \in T$, let $m_e := |\{v \in C \mid e \in P_{vz}\}|$ be the number of v-z paths that contain e. Let G'' be the graph, induced by the edges $e \in T$ with $m_e \geq \gamma/2$ ($G'' := \{z\}$ if no such edge exists). Moving from a leaf of T to the root z, the quantity m_e can only increase, hence the subgraph G'' is connected and $z \in V(G'')$. To upperbound $\phi_p(C, G'')$, we still use P_{vz} as v-z path, even if $\ell_{G''}(v, z)$ is attained by a different path. Consider any edge $e \in T$. If $e \in G'$,

then e contributes cost γc_e to $\phi_p(C, G')$, otherwise it contributes $2m_e c_e$. Note that $\gamma c_e \leq 2m_e c_e$ iff $m_e \geq \gamma/2$. By the definition of G'', the contribution of e to $\phi_p(C, G'')$ is $\min\{2m_e c_e, \gamma c_e\}$, which is never larger than the contribution to $\phi_p(C, G')$. The claim follows by applying the Core Detouring Theorem to the core G'':

$$E\Big[\sum_{v\in C} \ell(v, C' \cup \{z\})\Big] \leq \phi_p(C, G'') \leq \sum_{e\in T\setminus G''} 2m_e c_e + \sum_{e\in G''} \gamma c_e$$
$$= \sum_{e\in T} \min\{2m_e c_e, \gamma c_e\} \leq \sum_{e\in T\setminus G'} m_e c_e + \sum_{e\in G'} \gamma c_e = \phi_p(C, G'). \square$$

From Splittable to Unsplittable Flows. We first state the following simple lemma, which is implicitly given in [27].

Lemma 5. *[27] Let $c^*(x)$ be the minimum-cost of a cable installation supporting a capacity reservation $x = \{x_e\}_{e\in E}$. Then there is a polynomial-time computable concave function $g(\cdot)$ such that $c^*(x) \leq g(x) \leq 2\, c^*(x)$.*

The choice of $g(\cdot)$ here is $g(x) := \sum_{e\in E} c(e) f(x_e)$ with $f(z) := \min_{i=1,\ldots,k}\{\sigma_i + \delta_i \cdot z\}$ for $z > 0$ and $f(0) := 0$.

For an arbitrary concave cost function $g(\cdot)$, Karger and Minkoff [22] showed that there is always an optimum solution inducing a tree. It is not hard (just slightly technical) to turn their existential proof into a polynomial-time procedure to transform any given solution into a tree solution without increasing its cost with respect to $g(\cdot)$. The proof of Theorem 5, which is omitted here for lack of space, is obtained by combining that procedure with the concave function provided by Lemma 5.

The Splittable Case. Our algorithm `sssbb` for s-SSBB, which is described in Figure 2, is a slight variant of the algorithms in [12]. By adding dummy clients in the sink, we can assume that $|D|$ is a multiple of all the capacities μ_i. The quantity α is a proper constant to be fixed later. The *aggregation algorithm* [17] is a randomized procedure to aggregate a given set of demands $x(v) \in [0, U)$ on the nodes $v \in V(T)$ of a given tree T, under the assumption that the sum of the demands is a multiple of $U > 0$. This is obtained by moving demands over T such that: (1) The amount of flow along each edge of T is at most U, (2) The final demand $x'(v)$ at each node is either 0 or U, and (3) The expected demand at each node is preserved, that is: $\Pr[x'(v) = U] = x(v)/U$.

The algorithm initially selects a subset of k' cable types $i(1), i(2), \ldots, i(k')$. For notational convenience, we assume $\sigma_{i(k'+1)} = \infty$ and $i(0) = 0$. Then there is a sequence of $k' + 1$ rounds. In each round the demand of the clients (which is initially 1 for each client) is aggregated in a smaller and smaller subset of clients. At the beginning of round $t \geq 1$ the demand at each client is in $\{0, \mu_{i(t)}\}$. Each round t consists of three steps. Initially the demand is collected at a random subset of aggregation points (Collection Step). Then a Steiner tree is computed on the aggregation points, and the demand is aggregated along such tree via the aggregation algorithm in multiples of $\mu_{i(t+1)}$ (Aggregation Step). This is possible since the sum of the $d'_t(w)$'s, and hence of the $x(w)$'s, is a multiple of $\mu_{i(t+1)}$.

(S1) Select a subset of cable types $i(1), \ldots, i(k')$ in increasing order of capacity, where $i(1) = 1$ and $i(k') = k$.
(S2) For $t = 0, 1, \ldots, k'$:
(Collection) Let D_t be the set of nodes with positive demand. Each node in D_t is marked with probability $p_t = \alpha\sigma_{i(t)}/\sigma_{i(t+1)}$ (probability 1 if $t = 0$). Let D'_t be the set of marked nodes. Each node sends its demand to the closest node in $D'_t \cup \{r\}$ along a shortest path, using cables of type $i(t)$ (type 1 for $t = 0$). Let $d'_t(w)$ be the new demand collected at each $w \in D'_t \cup \{r\}$.
(Aggregation) If $t < k'$, compute a ρ_{st}-approximate Steiner tree T_t on $D'_t \cup \{r\}$. Apply the aggregation algorithm to T_t with $U = \mu_{i(t+1)}$ and $x(w) = d'_t(w) \pmod{\mu_{i(t+1)}}$ for each terminal node w. The corresponding flow is supported by installing cables of type $i(t+1)$ (at most one for each edge of T_t). Let $d''_t(w)$ be the new demand aggregated at each node w.
(Redistribution) If $t < k'$, for each node $w \in D'_t \cup \{r\}$, consider the subset of nodes $D_t(w) \subseteq D_t$ that sent their demand to w during the collection step (including w itself, if $w \neq r$). Uniformly select a random subset $\widetilde{D}_t(w)$ of $D_t(w)$ of cardinality $d''_t(w)/\mu_{i(t+1)}$. Send $\mu_{i(t+1)}$ units of flow back from w to each node in $\widetilde{D}_t(v)$ along shortest paths, installing cables of type $i(t+1)$.

Fig. 2. Algorithm `sssbb`

Eventually, the aggregated demand is redistributed back to the source nodes (Redistribution Step). Only cables of type $i(t)$ and $i(t+1)$ are used in round t. At the end of the round the demand at each client is in $\{0, \mu_{i(t+1)}\}$.

It remains to specify how the cable types $i(1), \ldots, i(k')$ are chosen. Differently from prior work on the topic, we use a randomized cable selection rule. Let $i(1) = 1$. Given $i(t)$, $1 < i(t) < k$, $i(t+1)$ is chosen as follows. Let $i'(t) > i(t)$ and $i''(t) > i(t)$ be the smallest indexes such that $\frac{\delta_{i'(t)}}{\delta_{i(t)}} \leq \frac{1}{\beta}$ and $\frac{\sigma_{i''(t)}}{\sigma_{i(t)}} \geq \beta$, respectively. Here $\beta > 1$ is a constant to be fixed later. If no proper $i'(t)$ (resp. $i''(t)$) exists, we let $i'(t) = k$ (resp. $i''(t) = k$). If $i'(t) \geq i''(t)$, $i(t+1) = i'(t)$. Otherwise, $i(t+1) = i''(t) - 1$ with probability $\frac{\sigma_{i''(t)} - \beta\sigma_{i(t)}}{\sigma_{i''(t)} - \sigma_{i''(t)-1}}$, and $i(t+1) = i''(t)$ otherwise. Note that, as required $i(1) = 1$ and $i(k') = k$. The proof of the following lemma will appear in the final version of the paper.

Lemma 6. *For any $t \in \{1, 2, \ldots, k'-2\}$ and $h \in \{0, 1, \ldots, k'-t-1\}$, and for any $s \in \{1, 2, \ldots, k\}$, $i(t') < s \leq i(t'+1)$:* **(a)** $\delta_{i(t+h)} \leq \frac{1}{\beta^h}\delta_{i(t)}$; **(b)** $E[\sigma_{i(t+h)}] \geq \beta^h E[\sigma_{i(t)}]$; **(c)** $E[\min\{\frac{\sigma_{i(t'+1)}}{\sigma_s}, \frac{\delta_{i(t')}}{\delta_s}\}] \leq \beta$.

Let A_t be the cost of the t-th round, $t \in \{0, 1, \ldots, k'\}$. Let moreover A^c_t, A^a_t, and A^r_t denote the collection, aggregation, and redistribution costs of the t-th round, $t \in \{1, \ldots, k'-1\}$ respectively. By $OPT(s)$ we denote the cost paid by the optimum solution for cables of type s. The proof of the following lemma is implicitly given in [12].

Lemma 7. *[12] For $t' \in \{1,\dots,k'\}$ and $t \in \{1,\dots,k'-1\}$:* **(1)** $\Pr[d \in D_{t'} | v \in D_0] = \frac{1}{\mu_{i(t')}}$; **(2)** $A_0 \leq \rho_{st} \sum_s \frac{\sigma_{i(1)}}{\sigma_s} OPT(s)$; **(3)** $E[A_{k'}] \leq \sum_s \frac{\delta_{i(k')}}{\delta_s} OPT(s)$; **(4)** $E[A_t^a] \leq E\Big[\sum_s \min\Big\{\rho_{st}\alpha\frac{\delta_{i(t)}}{\delta_s}, \rho_{st}\frac{\sigma_{i(t+1)}}{\sigma_s}\Big\} OPT(s)\Big]$; **(5)** $E[A_t^r] \leq E\Big[\frac{\delta_{i(t+1)}}{\delta_{i(t)}} A_t^c\Big]$.

Hence it remains to bound $E[A_t^c]$. Following [12,17], it is not hard to show that $E[A_t^c] \leq \frac{2}{\alpha} E[A_t^a]$. We next present an improved bound based on the Multi-Core Detouring Theorem. By adding dummy demands at the root, we can assume that $|D_t| = |D|/\mu_{i(t)} \gg p_t = \alpha\sigma_{i(t)}/\sigma_{i(t+1)}$ for all t, and consequently $|D_t|e^{-p_t|D_t|} + 0.8067/p_t \leq 0.8068/p_t$.

Lemma 8. *For $t = 1,\dots,k'-1$, $E[A_t^c] \leq E[\sum_s \min\{2\frac{\delta_{i(t)}}{\delta_s}, \frac{0.8068}{\alpha}\frac{\sigma_{i(t+1)}}{\sigma_s}\} OPT(s)]$.*

Proof. Let $j \in \{1,2,\dots,k\}$ be an integer value to be fixed later. We denote by G_j the graph induced by the edges where OPT installs at least one cable of type $s > j$. Note that this graph might be disconnected. By the Multi-Core Detouring Theorem applied to $C = D_t$, $z = r$, $p = p_t$ and $G' = G_j$, $E[A_t^c] := E[\sigma_{i(t)} \sum_{d\in D_t} \ell(d, D_t' \cup \{r\})] \leq E[\sigma_{i(t)}(2\sum_{d\in D_t} \ell_{G_j}(d,r) + \frac{0.8068}{p_t} c(G_j))]$.

By definition, $E\big[\frac{0.8068}{p_t}\sigma_{i(t)} c(G_j)\big] = E\big[\frac{0.8068\,\sigma_{i(t+1)}}{\alpha} c(G_j)\big] \leq E\big[\frac{0.8068}{\alpha}\sum_{s>j}\frac{\sigma_{i(t+1)}}{\sigma_s} OPT(s)\big]$. By Lemma 7.1, $\Pr[d \in D_t | d \in D] = \frac{1}{\mu_{i(t)}}$. Then $E[2\sigma_{i(t)} \sum_{d\in D_t} \ell_{G_j}(d,r)] = E[2\frac{\sigma_{i(t)}}{\mu_{i(t)}} \sum_{d\in D} \ell_{G_j}(d,r)] = E[2\delta_{i(t)} \sum_{d\in D} \ell_{G_j}(d,r)]$. Let $L_{t,j}$ be the cost of routing the flow as in OPT, but paying zero on the edges of G_j and $\delta_{i(t)}$ per unit of flow on the remaining edges. Then trivially $\delta_{i(t)} \sum_{d\in D} \ell_{G_j}(d,r) \leq L_{t,j}$. In turn, OPT pays at least δ_s per unit flow on each cable of type $s \leq j$, which implies $L_{t,j} \leq \sum_{s\leq j} \frac{\delta_{i(t)}}{\delta_s} OPT(s)$. We can conclude that $E[2\sigma_{i(t)} \sum_{d\in D_t} \ell_{G_j}(d,r)] \leq E[2\sum_{s\leq j} \frac{\delta_{i(t)}}{\delta_s} OPT(s)]$. Altogether $E[A_t^c] \leq E[2\sum_{s\leq j} \frac{\delta_{i(t)}}{\delta_s} OPT(s) + \frac{0.8068}{\alpha}\sum_{s>j} \frac{\sigma_{i(t+1)}}{\sigma_s} OPT(s)]$. Since, deterministically, $\delta_{i(t)}/\delta_s$ is decreasing in t while $\sigma_{i(t+1)}/\sigma_s$ is increasing in t, the claim follows by choosing j properly. □

The proof of Theorem 3, omitted here due to space constraints, follows easily by combining Lemmas 6, 7, and 8, and imposing $\beta = 2.80$ and $\alpha = 0.531$.

References

1. Becchetti, L., Könemann, J., Leonardi, S., Pál, M.: Sharing the cost more efficiently: improved approximation for multicommodity rent-or-buy. ACM Transactions on Algorithms 3(2) (2007)
2. Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L.: An Improved LP-based Approximation for Steiner tree. In: STOC (to appear, 2010) (Best Paper Award)
3. Duffield, N.G., Goyal, P., Greenberg, A., Mishra, P., Ramakrishnan, K.K., van der Merwe, J.E.: A flexible model for resource management in virtual private networks. In: SIGCOMM, pp. 95–108 (1999)
4. Eisenbrand, F., Grandoni, F.: An improved approximation algorithm for virtual private network design. In: SODA, pp. 928–932 (2005)

5. Eisenbrand, F., Grandoni, F., Oriolo, G., Skutella, M.: New approaches for virtual private network design. SIAM Journal on Computing 37(3), 706–721 (2007)
6. Eisenbrand, F., Grandoni, F., Rothvoß, T., Schäfer, G.: Approximating connected facility location problems via random facility sampling and core detouring. In: SODA, pp. 1174–1183 (2008)
7. Fingerhut, J.A., Suri, S., Turner, J.S.: Designing least-cost nonblocking broadband networks. Journal of Algorithms 24(2), 287–309 (1997)
8. Fleischer, L., Könemann, J., Leonardi, S., Schäfer, G.: Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In: STOC, pp. 663–670 (2006)
9. Garg, N., Gupta, A., Leonardi, S., Sankowski, P.: Stochastic analyses for online combinatorial optimization problems. In: SODA, pp. 942–951 (2008)
10. Garg, N., Khandekar, R., Konjevod, G., Ravi, R., Salman, F., Sinha, A.: On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design problem. In: Aardal, K., Gerards, B. (eds.) IPCO 2001. LNCS, vol. 2081, pp. 170–184. Springer, Heidelberg (2001)
11. Goyal, N., Olver, N., Shepherd, F.B.: The VPN conjecture is true. In: STOC, pp. 443–450 (2008)
12. Grandoni, F., Italiano, G.F.: Improved approximation for single-sink buy-at-bulk. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288, pp. 111–120. Springer, Heidelberg (2006)
13. Grandoni, F., Kaibel, V., Oriolo, G., Skutella, M.: A short proof of the VPN Tree Routing Conjecture on ring networks. Operations Research Letters 36(3), 361–365 (2008)
14. Guha, S., Meyerson, A., Munagala, K.: A constant factor approximation for the single sink edge installation problem. SIAM Journal on Computing 38(6), 2426–2442 (2009)
15. Gupta, A., Kleinberg, J., Kumar, A., Rastogi, R., Yener, B.: Provisioning a virtual private network: a network design problem for multicommodity flow. In: STOC, pp. 389–398 (2001)
16. Gupta, A., Kumar, A., Pal, M., Roughgarden, T.: Approximation via cost-sharing: simpler and better approximation algorithms for network design. Journal of the ACM 54(3), 11 (2007)
17. Gupta, A., Kumar, A., Roughgarden, T.: Simpler and better approximation algorithms for network design. In: STOC, pp. 365–372 (2003)
18. Gupta, A., Pál, M.: Stochastic Steiner trees without a root. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1051–1063. Springer, Heidelberg (2005)
19. Hurkens, C.A.J., Keijsper, J.C.M., Stougie, L.: Virtual Private Network Design: A Proof of the Tree Routing Conjecture on Ring Networks. SIAM Journal on Discrete Mathematics 21(2), 482–503 (2007)
20. Italiano, G.F., Leonardi, S., Oriolo, G.: Design of trees in the hose model: The balanced case. Operations Research Letters 34(6), 601–606 (2006)
21. Jothi, R., Raghavachari, B.: Improved approximation algorithms for the single-sink buy-at-bulk network design problems. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, pp. 336–348. Springer, Heidelberg (2004)
22. Karger, D.R., Minkoff, M.: Building steiner trees with incomplete global knowledge. In: FOCS, pp. 613–623 (2000)

23. Meyerson, A., Munagala, K., Plotkin, S.: Cost-distance: two metric network design. In: FOCS, pp. 624–630 (2000)
24. Rothvoß, T., Sanità, L.: On the complexity of the asymmetric VPN problem. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. LNCS, vol. 5687, pp. 326–338. Springer, Heidelberg (2009)
25. Swamy, C., Kumar, A.: Primal–dual algorithms for connected facility location problems. In: Jansen, K., Leonardi, S., Vazirani, V.V. (eds.) APPROX 2002. LNCS, vol. 2462, pp. 256–269. Springer, Heidelberg (2002)
26. Shmoys, D.B., Talwar, K.: A Constant Approximation Algorithm for the a priori Traveling Salesman Problem. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO 2008. LNCS, vol. 5035, pp. 331–343. Springer, Heidelberg (2008)
27. Talwar, K.: The single-sink buy-at-bulk LP has constant integrality gap. In: Cook, W.J., Schulz, A.S. (eds.) IPCO 2002. LNCS, vol. 2337, pp. 475–486. Springer, Heidelberg (2002)
28. van Zuylen, A.: Deterministic Sampling Algorithms for Network Design. In: Halperin, D., Mehlhorn, K. (eds.) ESA 2008. LNCS, vol. 5193, pp. 830–841. Springer, Heidelberg (2008)

Weak Completeness Notions for Exponential Time

Klaus Ambos-Spies and Timur Bakibayev

University of Heidelberg, Institut für Informatik,
INF 294, D-69120 Heidelberg, Germany

Abstract. Lutz [20] proposed the following generalization of hardness: While a problem A is hard for a complexity class C if all problems in C can be reduced to A, Lutz calls a problem weakly hard if a *nonnegligible* part of the problems in C can be reduced to A. For the exponential-time class E, Lutz formalized these ideas by introducing a resource-bounded (pseudo) measure on this class and by saying that a subclass of E is negligible if it has measure 0 in E.

Here we introduce and investigate new weak hardness notions for E, called E-*nontriviality* and *strong* E-*nontriviality*, which generalize Lutz's weak hardness notion for E and which are conceptually much simpler than Lutz's concept. Moreover, E-nontriviality may be viewed as the most general consistent weak hardness notion for E.

1 Introduction

The standard way for proving a problem to be intractable is to show that the problem is hard or complete for one of the standard complexity classes containing intractable problems. Lutz [20] proposed a generalization of this approach by introducing more general *weak* hardness notions which still imply intractability. While a set A is hard for a class C if *all* problems in C can be reduced to A (by a polynomial-time-bounded many-one reduction) and complete if it is hard and a member of C, Lutz proposed to call a set A weakly hard if a *nonnegligible* part of C can be reduced to A and to call A weakly complete if in addition $A \in$ C. For the exponential-time classes $\mathrm{E} = \mathrm{DTIME}(2^{lin})$ and $\mathrm{EXP} = \mathrm{DTIME}(2^{poly})$, Lutz formalized these ideas by introducing resource-bounded (Lebesgue) measures on these classes and by saying that a subclass of E is negligible if it has measure 0 in E (and similarly for EXP). A variant of these concepts, based on resource-bounded Baire category in place of measure, was introduced by Ambos-Spies [4] where now a class is declared to be negligible if it is meager in the corresponding resource-bounded sense.

A certain drawback of these weak hardness notions in the literature, called measure-hardness and category-hardness in the following, is that they are based on the somewhat technical concepts of resource-bounded measure and resource-bounded category, respectively. So here we introduce some alternative weak hardness notions which are conceptually much simpler and are solely based on the basic concepts of computational complexity theory.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 503–514, 2010.

While the weak hardness notions in the literature implicitly used the fact that, by the time-hierarchy theorem, the linear-exponential time class E is actually a (proper) hierarchy, where the individual levels are the deterministic-time classes $\mathrm{E}_k = \mathrm{DTIME}(2^{kn})$, our primary new weak hardness notion for E, called E-*nontriviality*, is based on this observation explicitly. We consider a subclass of E to be negligible, if it is contained in a fixed level E_k of the linear-exponential-time hierarchy. In other words, a set A is E-nontrivial if it has predecessors (under p-m-reducibility) from arbitrarily high levels of this hierarchy.

Since any level E_k of E has measure 0 in E and is meager in E, E-nontriviality generalizes the previously introduced weak hardness notions for E. On the other hand, since $\mathrm{P} \subseteq \mathrm{E}_1$, E-nontriviality still guarantees intractability. In fact, we may argue that E-nontriviality is the most general weak hardness notion for E if for such a notion we do not only require that it generalizes E-hardness and implies intractability but that it also reflects the internal, hierarchical structure of E.

The second new weak hardness notions we will consider, called *strong* E-*nontriviality*, may be viewed as a link between the weak notion of E-nontriviality and the stronger notions of measure- and category-hardness. A strongly E-nontrivial set does not only have predecessors from arbitrarily high levels $\mathrm{E} \setminus \mathrm{E}_k$ of the hierarchy E but, for any given $k \geq 1$, it has a predecessor in E which is *almost-everywhere* complex for the kth level of the linear exponential-time hierarchy, i.e., which is E_k-bi-immune.

The outline of the paper is as follows.

After formally introducing our new weak hardness notions for E - E-nontriviality and strong E-nontriviality - in Section 2, in Section 3 we give some examples of intractable but still E-trivial sets in E thereby showing that E-nontriviality is a strict refinement of intractability. First we observe that sets of sufficiently low hyper-polynomial complexity are E-trivial. Though this observation is not surprising, it gives us some first nontrivial facts on the distribution of trivial and nontrivial sets in E. For instance it implies that the only sets which code all E-trivial sets in E are the E-hard sets. We then show (what might be more surprising) that there are E-trivial sets in E of arbitrarily high complexity, i.e., that, for any $k \geq 1$, there is an E-trivial set in $\mathrm{E} \setminus \mathrm{E}_k$. In fact, by generalizing a result of Buhrman and Mayordomo [13] for measure hardness, we obtain some natural examples of such sets by showing that the sets of the Kolmogorov-random strings w.r.t. exponential-time-bounded computations are E-trivial.

In Section 4 we give some examples of (strongly) E-nontrivial sets and prove a hierarchy theorem for the weak E-completeness notions. In fact, we show that the complete sets under the weak hardness notions considered here can be distinguished by their minimum densities. So there are exptally sets which are E-nontrivial whereas no such set can be strongly E-nontrivial, and there are tally sets which are strongly E-nontrivial whereas no tally set is category-hard for E. (And, as shown in the literature already, there are sparse category-hard sets for E whereas every measure-hard set for E is exponentially dense.)

In Section 5 we analyse the information content of weakly complete sets thereby giving some more structural differences among the complete sets under the various weak hardness notions. For instance we show, that the disjoint union of two E-trivial sets is E-trivial again, and that E-trivial sets *don't help*, i.e., if an E-hard set H can be reduced to the disjoint union of sets A and B where A is E-trivial then H can be reduced to B already. In other words, if we decompose an E-complete set into two incomplete parts then both parts are E-nontrivial. For the other weak hardness notions the corresponding claims fail.

Finally, in Section 6 we give a short summary of results on some other aspects of our new weak hardness notions which will be presented in more details somewhere else.

Our notation is standard (see e.g. the monographs of Balcázar et al. [10] and [11]). Due to lack of space many proofs are omitted and where a proof is presented usually only a sketch or even only a hint to the ideas underlying the proof is given.

2 E-Nontriviality and Strong E-Nontriviality

We start with some notation. The exponential time classes we will deal with are the classes

$$\mathrm{E} = \bigcup_{k\geq 1} \mathrm{DTIME}(2^{kn}) \qquad (\textit{Linear Exponential Time}) \tag{1}$$

$$\mathrm{EXP} = \bigcup_{k\geq 1} \mathrm{DTIME}(2^{n^k}) \qquad (\textit{Polynomial Exponential Time}) \tag{2}$$

where we will use the following abbreviations for the individual levels of these classes:

$$\mathrm{E}_k = \mathrm{DTIME}(2^{kn}) \text{ and } \mathrm{EXP}_k = \mathrm{DTIME}(2^{n^k}).$$

Note that, by the time-hierarchy theorem, the hierarchies of the linear-exponential-time classes and of the polynomial-exponential-time classes are proper, and that

$$\mathrm{E}_1 = \mathrm{EXP}_1 \subset \mathrm{E} \subset \mathrm{EXP}_2.$$

For comparing problems we use the polynomial-time-bounded version of many-one reducibility (p-m-reducibility for short) where a set A is p-m-reducible to a set B ($A \leq^p_m B$) via f if f is polynomial-time computable and $A(x) = B(f(x))$ for all strings x. We let $\mathrm{P}_m(A) = \{B : B \leq^p_m A\}$ denote the class of predecessors of A under p-m-reducibility and we say that A *codes* a class C of problems if $\mathrm{C} \subseteq \mathrm{P}_m(A)$.

2.1 Nontriviality

Definition 1. *A set A is* trivial for E *(or* E-trivial *for short) if*

$$\exists\, k \geq 1\ (\mathrm{P}_m(A) \cap \mathrm{E} \subseteq \mathrm{E}_k) \tag{3}$$

holds, and A is nontrivial for E *(or* E-nontrivial *for short) otherwise.*

Note that the class of the E-nontrivial sets is closed upwards under $\leq_m^p$ hence, in particular, closed under p-m-equivalence. Also note that for an E-nontrivial set A there are infinitely many numbers $k \geq 1$ such that A has a predecessor in $\mathrm{E}_{k+1} \setminus \mathrm{E}_k$. In fact, for any E-nontrivial set A,

$$\forall\, k \geq 1\ \exists B \in \mathrm{E}_{k+1} \setminus \mathrm{E}_k\ (B \leq_m^p A) \tag{4}$$

holds. This is an easy consequence of the following variant of the padding lemma.

Lemma 1. *Let A and $k \geq 1$ be given such that $A \in \mathrm{E}_{k+1} \setminus \mathrm{E}_k$. Then, for any $k' \leq k$ (with $k' \geq 1$), there is a set $A' \in \mathrm{E}_{k'+1} \setminus \mathrm{E}_{k'}$ such that $A' =_m^p A$.*

Proof (Idea). Given $k \geq 2$ and $A \in \mathrm{E}_{k+1} \setminus \mathrm{E}_k$, let $A' = \{0^{f(|x|)}1x : x \in A\}$ for $f(n) = \lfloor \frac{n}{k} \rfloor$. Then $A' =_m^p A$ and $A' \in \mathrm{E}_k \setminus \mathrm{E}_{k-1}$. The claim follows by induction.

2.2 Strong Nontriviality

Recall that a set A is *almost everywhere (a.e.) $t(n)$-complex* if, for any Turing machine M computing A, the runtime $time_M(x)$ of M on input x exceeds $t(|x|)$ for almost all (i.e., all but finitely many) strings x. Almost everywhere complexity coincides with bi-immunity, i.e., a set A is a.e. $t(n)$-complex if and only if A is DTIME($t(n)$)-bi-immune (see Balcázar et al. [11]). Here a set A is C-bi-immune for a class C if there is no infinite set $B \in \mathrm{C}$ such that $B \subseteq A$ or $B \cap A = \emptyset$. In the following we will tacitly use the fact that, by the time-hierarchy theorem for a.e. complexity by Geske et al. [14], there are E_k-bi-immune sets $A \in E_{k+1}$ (for any $k \geq 1$).

Definition 2. *A set A is* strongly nontrivial for E *(or* strongly E-nontrivial *for short) if*

$$\forall\, k \geq 1\ \exists\, B\ (B \in \mathrm{P}_m(A) \cap \mathrm{E}\ \&\ B\ \mathrm{E}_k\text{-}bi\text{-}immune) \tag{5}$$

holds; and A is weakly trivial for E *(or* weakly E-trivial *for short) otherwise.*

Note that, for any E_k-bi-immune set B, $B \notin \mathrm{E}_k$. So, any strongly E-nontrivial set A is E-nontrivial. Moreover, just as a set A is E-nontrivial if and only if, for any $k \geq 1$ there is a predecessor B of A which is infinitely often 2^{kn}-complex but not infinitely often $2^{(k+1)n}$-complex (see (4) above), we can show that a set A is strongly E-nontrivial if and only if, for any $k \geq 1$ there is a predecessor B of A which is a.e. 2^{kn}-complex but not a.e. $2^{(k+1)n}$-complex:

$$\forall\, k \geq 1\ \exists B \in \mathrm{E} \cap \mathrm{P}_m(A)\ (B\ \mathrm{E}_k\text{-bi-immune}\ \&\ B\ \text{not}\ \mathrm{E}_{k+1}\text{-bi-immune}) \tag{6}$$

Of great technical interest is the following alternative characterization of strong E-nontriviality.

Theorem 1 (Characterization Theorem for Strong Nontriviality). *A set A is strongly* E*-nontrivial if and only if there is an* E_1*-bi-immune set $B \in \mathrm{E}$ such that $B \leq_m^p A$.*

The nontrivial direction of Theorem 1 follows from the following lemma by considering the sets B_k there (for a given E_1-bi-immune predecessor $B \in \mathrm{E}$ of A).

Lemma 2. *Let B be E_1-bi-immune. Then, for any $k \geq 1$, there is an E_k-bi-immune set B_k and an EXP_k-bi-immune set B'_k such that $B_k, B'_k \in \mathrm{P}_m(B)$. If moreover $B \in \mathrm{E}$ then the set B_k can be chosen such that $B_k \in \mathrm{P}_m(B) \cap \mathrm{E}$.*

Proof (Idea). The idea is borrowed from Ambos-Spies et al. [9] where a similar lemma for randomness in place of bi-immunity is proven: For some appropriately defined padding functions, the sets B_k and B'_k are chosen so that they consist of those strings for which the padded versions are members of B, namely, $B_k = \{x : 0^{k|x|}1x \in B\}$ and $B'_k = \{x : 0^{|x|^k}1x \in B\}$. Since B is *almost everywhere* complex, hence, in particular, complex on the set of padded strings, the sets B_k and B'_k are a.e. complex again. Moreover, due to the compression of strings, the complexity is increased by the required amount.

3 Some Examples of E-Trivial Sets in E

In order to show that E-nontriviality does not coincide with intractability, here we give some examples of intractable but E-trivial sets. As one might expect, sets of sufficiently low time complexity are E-trivial. As we will also show, however, E-trivial sets can be found at all levels of the linear-exponential hierarchy.

3.1 Sets of Low Complexity Are Trivial

Lemma 3. *Let t be a nondecreasing, time constructible function such that, for some number $k \geq 1$,*

$$t(p(n)) \leq_{a.e.} 2^{kn} \tag{7}$$

for all polynomials p. Then any set $A \in \mathrm{DTIME}(t(n))$ is E-trivial.

Proof (Sketch). Given $A \in \mathrm{DTIME}(t(n))$ it follows from (7) that $\mathrm{P}_m(A) \subseteq \mathrm{E}_k$. So A is E-trivial.

Theorem 2. *There is an E-trivial set $A \in \mathrm{E} \setminus \mathrm{P}$.*

Proof (Sketch). Note that, for any polynomial p, $p(n) \leq_{a.e.} 2^{(\log n)^2}$ and that $2^{(\log n)^4} \notin O(2^{(\log n)^2} \cdot \log(2^{(\log n)^2}))$. So $\mathrm{P} \subseteq \mathrm{DTIME}(2^{(\log n)^2}) \subset \mathrm{DTIME}(2^{(\log n)^4})$ (where the strictness of the latter inclusion holds by the time-hierarchy theorem). Moreover, $2^{(\log\ p(n))^4} \leq_{a.e.} 2^n$ for all polynomials p. So, by the former, there is a set $A \in \mathrm{DTIME}(2^{(\log n)^4}) \setminus \mathrm{P}$ and, by the latter and by Lemma 3, A is E-trivial.

Theorem 2 can be strengthened by using some results on the p-m-degrees of hyperpolynomial shifts proven in Ambos-Spies [1]. A set $A_h = \{1^{h(|x|)}0x : x \in A\}$ is called a hyperpolynomial shift of A if h is a time constructible, nondecreasing function $h : \mathbb{N} \to \mathbb{N}$ such that h dominates all polynomials. Note that, for any hyperpolynomial shift A_h of a set $A \in \mathrm{EXP}$, $A_h \in \mathrm{DTIME}(t(n))$ for a function

$t(n)$ as in Lemma 3 whence A_h is E-trivial. Now, in [1] it has been shown that for any computable sets A and B such that $A \not\leq_m^p B$ there is a hyperpolynomial shift A_h of A such that $A_h \not\leq_m^p B$. By letting A be any E-complete set, the above implies:

Theorem 3. *For any computable set B which is not* E*-hard there is an* E*-trivial set T such that $T \not\leq_m^p B$.*

So the only sets in E which code all E-trivial sets in E are the E-complete sets.

3.2 Trivial Sets of High Complexity

Having given examples of intractable E-trivial sets of low hyperpolynomial complexity, we now show that there are E-trivial sets at arbitrarily high levels $\mathrm{E} \setminus \mathrm{E}_k$ of the E-hierarchy. (So, by Lemma 1, there are E-trivial sets at all levels $\mathrm{E}_{k+1} \setminus \mathrm{E}_k$ of the E-hierarchy.)

Theorem 4. *For any $k \geq 1$ there is an* E*-trivial set A in $\mathrm{E} \setminus \mathrm{E}_k$.*

The proof of Theorem 4 is based on the following straightforward observation.

Lemma 4 (Boundedness Lemma). *Let A and B be sets and let f be a p-m-reduction function such that $A \in \mathrm{E}_k$, $B \leq_m^p A$ via f, and*

$$\forall^\infty x \; (|f(x)| \leq k' \cdot |x| + k'' \quad or \quad f(x) \notin A) \tag{8}$$

(for some $k, k', k'' \geq 1$). Then $B \in \mathrm{E}_{k' \cdot k}$.

Proof (of Theorem 4; sketch). Fix $k \geq 1$ and let $\{E_e^k : e \geq 0\}$ and $\{f_e : e \geq 0\}$ be enumerations of E_k and of the class of the p-m-reduction functions, respectively, such that $E_e^k(x)$ can be computed in time $O(2^{(k+1)max(e,|x|)})$ and $f_e(x)$ can be computed in time $O(2^{max(e,|x|)})$ (uniformly in e and x).

By a diagonal argument we define a set $A \in \mathrm{E}_{k+2}$ which meets the requirements

$$\Re_{2e} : A \neq E_e^k$$

and

$$\Re_{2e+1} : \forall x \in \Sigma^* \; (|f_e(x)| > |x| + e + 1 \Rightarrow f_e(x) \notin A)$$

for $e \geq 0$.

Obviously, the requirements with even indices ensure that $A \notin \mathrm{E}_k$. Similarly, by $A \in \mathrm{E}_{k+2}$, the requirements with odd indices ensure that A is E-trivial since, by Lemma 4, $\mathrm{P}_m(A) \subseteq \mathrm{E}_{2(k+2)}$.

For the definition of A, call a string y *forbidden* if $y = f_e(x)$ for some number e and some string x such that $|x| + e + 1 < |y|$. Note that the requirements $\Re_{2e+1}$ are met if we do not put any forbidden string into A. Moreover, the question whether a string y is forbidden can be decided in $O(2^{2|y|})$ steps. Finally, by a simple counting argument, for any $n \geq 0$ there is a string of length n which is not forbidden.

So if we let $A = \{y_e : y_e \notin E_e^k\}$ where y_e is the least string of length e which is not forbidden then all requirements are met and, as one can easily check, $A \in \mathrm{E}_{k+2}$.

Alternatively we obtain E-trivial sets of high complexity in E by refining a result of Buhrman and Mayordomo [13] on the random strings in the setting of time-bounded Kolmogorov complexity. Call a string x $t(n)$-K-random if there is no $t(n)$-time-bounded Turing machine compressing x (for the formal definition, see e.g. Li and Vitanyi [18]), and let R^t be the set of $t(n)$-K-random strings. Buhrman and Mayordomo have shown that, for an exponential time bound $t(n) = 2^{kn}$ $(k \geq 2)$, the set R^t of $t(n)$-K-random strings is not weakly E-complete in the sense of Lutz. This can be strengthened as follows.

Theorem 5. *For $t(n) = 2^{kn}$ $(k \geq 2)$, the set R^t of the $t(n)$-K-random strings is* E*-trivial.*

Proof (Idea). As one can easily check, $R^{2^{kn}} \in$ E. So, given a set A and a p-m-reduction function f such that $A \leq_m^p R^{2^{kn}}$ via f, by Lemma 4 it suffices to show that for almost all x such that $|f(x)| > 2|x|$, $f(x) \notin R^{2^{kn}}$. But this is straightforward, since for a string x with $|f(x)| > 2|x|$, $f(x)$ can be computed from the shorter string x in polynomial time hence, for sufficiently large x, in time $2^{k|x|}$.

Remark. In this paper we only look at E-(non)trivial sets in E, i.e., investigate (strong) E-nontriviality as a weak completeness notion, and do not consider the corresponding, more general weak hardness notion. So we only remark here that outside of E we can find computable E-trivial sets of arbitrarily high time complexity. Moreover, there are numerous noncomputable E-trivial sets. In fact, the class of E-trivial sets has (classical) Lebesgue measure 1. These observations immediately follow from results in the literature about p-m-minimal pairs. (Sets A and B form a p-m-minimal pair if, for any set C such that $C \leq_m^p A$ and $C \leq_m^p B$, $C \in$ P.) It suffices to observe that, for sets A and B such that B is E-hard and A and B form a minimal pair, A is E-trivial.

4 On the Density of E-Nontrivial and Strongly E-Nontrivial Sets

Lutz [20] has shown that any E-measure complete set is exponentially dense whereas for E-category completeness introduced by Ambos-Spies [4] there are sparse E-category complete sets. (In fact, in [4] weak completeness notions were introduced for various time-bounded category concepts. Here we refer to the category concept called AFH-category there which proved to be useful for analysing time-bounded measure (see [7] and [9] for more details) since - in contrast to the classical Baire category concept - this concept is compatible with measure.) By results in [4] and [9], however, E-category complete sets cannot be tally.

4.1 A Tally Strongly E-Nontrivial Set

So, in order to distinguish strong E-nontriviality from E-category completeness (and E-measure completeness), it suffices to show that there are tally strongly E-nontrivial sets in E. To do so we need the following straightforward observation.

Lemma 5. *Let $A \in \mathrm{E}$ be E_{k+1}-bi-immune ($k \geq 1$) and let $\hat{A}$ be the length language $\hat{A} = \{x : 0^{|x|} \in A\}$. Then $\hat{A} \in \mathrm{E}$, $\hat{A} \leq_m^p A$, and $\hat{A}$ is E_k-bi-immune.*

Theorem 6. *There is a tally set $A \in \mathrm{E}$ which is strongly* E*-nontrivial.*

Proof. By Lemma 5 there is a length language $A_1 \in \mathrm{E}$ which is E_1-bi-immune. Moreover, by Theorem 1, A_1 is strongly E-nontrivial. Since, for the tally set $A = A_1 \cap \{0\}^*$, A is in E and A is p-m-equivalent to A_1, it follows from p-m-invariance of strong E-nontriviality that A has the desired properties.

4.2 A Lower Bound on the Density of Strongly Nontrivial Sets

In order to distinguish E-nontriviality from strong E-nontriviality we look at very sparse sets. A set A is *exptally* if $A \subseteq \{0^{\delta(n)} : n \geq 0\}$ where $\delta : \mathbb{N} \to \mathbb{N}$ is the *iterated exponential function* inductively defined by $\delta(0) = 0$ and $\delta(n+1) = 2^{\delta(n)}$.

We first observe that no exptally set in E is strongly E-nontrivial.

Theorem 7. *Let $A \in \mathrm{E}$ be exptally. Then A is not strongly* E*-nontrivial.*

Since any strongly E-nontrivial set has an E_1-bi-immune (hence P-bi-immune) predecessor, it suffices to show the following.

Lemma 6. *Let A and B be sets such that $A \in \mathrm{E}$, A is exptally, and $B \leq_m^p A$. Then B is not* P*-bi-immune.*

The idea of the proof of Lemma 6 is as follows. Given a polynomial bound p for a p-m-reduction f from B to A, let $D = \{0^{\delta'(n)} : n \geq 0\}$ where $\delta'(n)$ is the least number m such that $p(m+1) \geq \delta(n)$. Then D is infinite and polynomial-time computable. Moreover, for $x \in D$, $B(x)$ can be computed in polynomial time using the reduction $B(x) = A(f(x))$. Namely if, for $x = 0^{\delta'(n)}$, $f(x) = 0^{\delta(r)}$ then $r < n$ whence, by definition of δ and δ' and by $A \in \mathrm{E}$, $A(f(x))$ can be computed in $poly(|x|)$ steps; and if $f(x)$ is not of the form $0^{\delta(r)}$, then $B(x) = A(f(x)) = 0$.

The observation that exptally sets in E cannot be strongly E-nontrivial can be generalized as follows.

Theorem 8. *Let $A \in \mathrm{E}$ and assume that there is an infinite polynomial-time computable set $B \subseteq \{0\}^*$ such that, for any number n with $0^n \in B$,*

$$A \cap \{x : n \leq |x| < 2^n\} = \emptyset. \tag{9}$$

Then A is not strongly E*-nontrivial.*

Theorem 8 implies that many constructions (of sets in E) in the theory of the polynomial-time degrees which are based on so-called gap languages (see e.g. Section 3 of [3]) yield sets which are not strongly E-nontrivial.

4.3 Exptally Sets and Nontriviality

In contrast to the above negative result on strong E-nontriviality for exptally sets, there are exptally sets in E which are E-nontrivial. In fact, *any* sufficiently complex exptally set $A \in \mathrm{E}$ is E-nontrivial.

Theorem 9. *Let $A \in \mathrm{E} \setminus \mathrm{E}_1$ be exptally. Then A is* E*-nontrivial.*

Proof (Idea). Given $k \geq 1$, we have to show that there is a set $A_k \leq^p_m A$ such that $A_k \in \mathrm{E} \setminus \mathrm{E}_k$. Such a set A_k is obtained from A by compressing strings of the form $0^{\delta(n)}$ by the factor k: $A_k = \{0^{\lfloor \frac{\delta(n)}{k} \rfloor} : 0^{\delta(n)} \in A\}$. Note that, by A being exptally, all (but a finite amount) information about A is coded into A_k in k-compressed form. This easily implies the claim.

Note that, by a straightforward diagonalization, there is an exptally set $A \in \mathrm{E} \setminus \mathrm{E}_1$. So, by Theorems 9 and 7, there is an E-nontrivial set in E which is not strongly E-nontrivial.

4.4 The Hierarchy of Weak Completeness Notions

By the above given differences in the possible densities of the sets with the various weak completeness properties we immediately get the following hierarchy theorem.

Theorem 10. *For any set A the following hold.*

$$\begin{array}{c} A \text{ E-}hard \\ \Downarrow \\ A \text{ E-}measure\ hard \\ \Downarrow \\ A \text{ E-}category\ hard \\ \Downarrow \\ A\ strongly \text{ E-}nontrivial \\ \Downarrow \\ A \text{ E-}nontrivial \\ \Downarrow \\ A\ intractable \end{array} \tag{10}$$

Moreover all implications are strict and witness sets A for strictness can be found in E*.*

5 On the Information Content of E-Nontrivial and Strongly E-Nontrivial Sets

In the preceding section we have distinguished E-nontriviality from the stronger weak completeness notions for E by analysing the possible densities of sets with these properties. Here we present another difference in the sense of information

content. We look at the following question: If we split a (weakly) complete set A into two parts A_0 and A_1, is at least one of the parts (weakly) complete again? As we will show, for E-nontriviality the answer is YES whereas for the other weak completeness notions the answer is NO.

In order to make our question more precise we need the following notion. A splitting of a set A into two disjoint sets A_0 and A_1 is a *p-splitting* if there is a set $B \in \mathrm{P}$ such that $A_0 = A \cap B$ and $A_1 = A \cap \overline{B}$. Note that for a p-splitting (A_0, A_1) of A, $A_0, A_1 \leq_m^p A$ and $A \equiv_m^p A_0 \oplus A_1$ (where $A_0 \oplus A_1$ is the effective disjoint union $\{0x : x \in A_0\} \cup \{1y : y \in A_1\}$ of A_0 and A_1).

Now Ladner [17] has shown that any computable intractable set A can be p-split into two lesser intractable problems, i.e., into problems $A_0, A_1 \notin \mathrm{P}$ such that $A_0, A_1 <_m^p A$. So, in particular, any E-complete set A can be p-split into two incomplete sets. In fact, by analysing Ladner's proof, the set $B \in \mathrm{P}$ defining the p-splitting is a gap language. So, by Theorem 8, we obtain the following stronger observation from Ladner's proof.

Lemma 7. *Let $A \in \mathrm{E} \setminus \mathrm{P}$. There is a p-splitting of A into sets $A_0, A_1 \notin \mathrm{P}$ such that A_0 and A_1 are weakly* E*-trivial.*

So, in particular, any E-complete (E-measure complete, E-category complete, strongly E-nontrivial) set A has a p-splitting into sets A_0 and A_1 which are not E-complete (E-measure complete, E-category complete, strongly E-nontrivial). For E-nontriviality, however, the corresponding fact fails.

Theorem 11. *Let A be* E*-nontrivial and let (A_0, A_1) be a p-splitting of A. Then A_0 is* E*-nontrivial or A_1 is* E*-nontrivial (or both).*

The key to the proof is the simple observation that, for a p-splitting C_0, C_1 of a set $C \notin \mathrm{E}_k$, $C_0 \notin \mathrm{E}_k$ or $C_1 \notin \mathrm{E}_k$.

Proof (Sketch). For a contradiction assume that A_0 and A_1 are E-trivial. Fix k_i such that $\mathrm{P}_m(A_i) \cap \mathrm{E} \subseteq \mathrm{E}_{k_i}$ $(i = 0, 1)$ and let $k = \max(k_0, k_1)$. Moreover, fix $B \in \mathrm{P}$ such that $A_0 = A \cap B$ and $A_1 = A \cap \overline{B}$. Finally, by E-nontriviality of A, fix a set $C \in \mathrm{E} \setminus \mathrm{E}_k$ such that $C \leq_m^p A$ and let f be a polynomial-time computable function such that $C \leq_m^p A$ via f. Then, for $D = \{x : f(x) \in B\}$, $D \in \mathrm{P}$. Now, consider the p-splitting $C_0 = C \cap D$ and $C_1 = C \cap \overline{D}$ of C given by D. Then $C_0, C_1 \in \mathrm{E}$ and, as one can easily check, $C_0 \leq_m^p A_0$ and $C_1 \leq_m^p A_1$. Since, by the above observation, $C_0 \notin \mathrm{E}_k$ or $C_1 \notin \mathrm{E}_k$, this contradicts the choice of k.

For an E-complete set A we obtain the following interesting variant of Theorem 11, which says that for a proper splitting of an E-complete set *both* parts are E-nontrivial.

Theorem 12. *Let A be* E*-complete and let (A_0, A_1) be a p-splitting of A such that $A_0, A_1 <_m^p A$. Then A_0 and A_1 are* E*-nontrivial.*

We omit the quite involved proof which requires some new result on the distribution of the E_k-bi-immune sets in E.

6 Further Results and Open Problems

We conclude with a short summary of some other results on our new weak completeness notions which will appear somewhere else.

6.1 Comparing Weak Hardness for E and EXP

While, by a simple padding argument, E-hardness and EXP-hardness coincide, Juedes and Lutz [16] have shown that E-measure hardness implies EXP-measure hardness whereas the converse in general fails. Moreover, by using similar ideas, the corresponding results have been obtained for category hardness (see [4]).

Now we can easily adapt the concepts of (strong) nontriviality for E to the polynomial-exponential time class EXP by replacing E and E_k in the definitions by EXP and EXP_k, respectively. Then, the arguments of [16] can be easily duplicated to show that strong E-nontriviality implies strong EXP-nontriviality but in general not vice versa.

For clarifying the relations between E-nontriviality and EXP-nontriviality, however, the above arguments fail and new much more sophisticated techniques have to be employed. As it turns out, in contrast to the above results, E-nontriviality and EXP-nontriviality are independent. I.e. neither E-nontriviality implies EXP-nontriviality nor EXP-nontriviality implies E-nontriviality. For details see [5].

6.2 Weak Hardness under Weak Reducibilities

The classical approach to generalize hardness notions is to generalize (weaken) the reducibilities underlying the hardness concepts, i.e., to allow more flexible codings in the reductions. So Watanabe [22] has shown that weaker reducibilities than p-m-reducibility like p-btt-reducibility (bounded truth-table), p-tt-reducibility (truth-table) and p-T-reducibility (Turing) yield more E-complete sets. For measure-completeness and category-completeness similar results have been shown in [12] (for both E and EXP). For strong nontriviality we obtain the corresponding results, i.e., a complete separation of p-m, p-btt, p-tt, and p-T (for both E and EXP), by fairly standard methods. For nontriviality, however, the separations of E-nontriviality under p-m, p-btt, p-tt, and p-T reducibilities require some quite involved and novel speed-up diagonalization technique. The reason why standard methods fail in this setting might be explained by the fact that - in contrast to the above results - EXP-nontriviality under p-m, p-btt, and p-tt coincide. See [6] for details.

References

1. Ambos-Spies, K.: Honest polynomial time reducibilities and the P = ?NP problem. J. Comput. System Sci. 39, 250–281 (1989)
2. Ambos-Spies, K.: Minimal pairs for polynomial time reducibilities. In: Börger, E. (ed.) Computation Theory and Logic. LNCS, vol. 270, pp. 1–13. Springer, Heidelberg (1987)

3. Ambos-Spies, K.: Polynomial time reducibilities and degrees. Stud. Logic Found. Math. 140, 683–705 (1999)
4. Ambos-Spies, K.: Resource-bounded genericity. London Math. Soc. Lecture Note Ser. 224, 1–59 (1996)
5. Ambos-Spies, K., Bakibayev, T.: Comparing nontriviality for E and EXP. Submitted for publication
6. Ambos-Spies, K., Bakibayev, T.: Nontriviality for exponential time w.r.t. weak reducibilities. In: Proceedings of TAMC (to appear, 2010)
7. Ambos-Spies, K., Mayordomo, E.: Resource-bounded measure and randomness. Lecture Notes in Pure and Appl. Math. 187, 1–47 (1997)
8. Ambos-Spies, K., Neis, H.-C., Terwijn, S.A.: Genericity and measure for exponential time. Theoret. Comput. Sci. 168, 3–19 (1996)
9. Ambos-Spies, K., Terwijn, S.A., Zheng, X.: Resource bounded randomness and weakly complete problems. Theoret. Comput. Sci. 172, 195–207 (1997)
10. Balcázar, J.L., Díaz, J., Gabarró, J.: Structural complexity, 2nd edn., vol. I. Springer, Berlin (1995)
11. Balcázar, J.L., Díaz, J., Gabarró, J.: Structural complexity, vol. II. Springer, Berlin (1990)
12. Ambos-Spies, K., Mayordomo, E., Zheng, X.: A Comparison of Weak Completeness Notions. In: Proceedings of the 11th Annual IEEE Conference on Computational Complexity, pp. 171–178 (1996)
13. Buhrman, H., Mayordomo, E.: An excursion to the Kolmogorov random strings. J. Comput. System Sci. 54, 393–399 (1997)
14. Geske, J.G., Huỳnh, D.T., Selman, A.L.: A hierarchy theorem for almost everywhere complex sets with application to polynomial complexity degrees. In: Brandenburg, F.J., Wirsing, M., Vidal-Naquet, G. (eds.) STACS 1987. LNCS, vol. 247, pp. 125–135. Springer, Heidelberg (1987)
15. Juedes, D.W., Lutz, J.H.: The complexity and distribution of hard problems. SIAM J. Comput. 24, 279–295 (1995)
16. Juedes, D.W., Lutz, J.H.: Weak completeness in E and E_2. Theoret. Comput. Sci. 143, 149–158 (1995)
17. Ladner, R.E.: On the structure of polynomial time reducibility. J. Assoc. Comput. Mach. 22, 155–171 (1975)
18. Li, M., Vitányi, P.: An introduction to Kolmogorov complexity and its applications. In: Graduate Texts in Computer Science, 2nd edn., pp. xx+637, Springer, New York (1997)
19. Lutz, J.H.: Almost everywhere high nonuniform complexity. J. Comput. System Sci. 44, 220–258 (1992)
20. Lutz, J.H.: Weakly hard problems. SIAM J. Comput. 24, 1170–1189 (1995)
21. Mayordomo, E.: Almost every set in exponential time is P-bi-immune. Theoret. Comput. Sci. 136, 487–506 (1994)
22. Watanabe, O.: A comparison of polynomial time completeness notions. Theoret. Comput. Sci. 54, 249–265 (1987)
23. Lutz, J.H.: Category and measure in complexity classes. SIAM J. Comput. 19, 1100–1131 (1990)
24. Lutz, J.H.: The quantitative structure of exponential time. Complexity theory retrospective, vol. II, pp. 225–260 (1997)

Efficient Evaluation of Nondeterministic Automata Using Factorization Forests*

Mikołaj Bojańczyk and Paweł Parys

University of Warsaw

Abstract. In the first part of the paper, we propose an algorithm which inputs an NFA $\mathcal{A}$ and a word $a_1 \cdots a_n$, does a precomputation, and then answers queries of the form: "is the infix $a_i \cdots a_j$ accepted by $\mathcal{A}$?". The precomputation is in time $\text{poly}(\mathcal{A}) \cdot n$, and the queries are answered in time $\text{poly}(\mathcal{A})$. This improves on previous algorithms that worked with the exponentially less succinct DFA's or monoids.

In the second part of the paper, we propose a transducer model for data trees. We show that the transducer can be evaluated in linear time. We use this result to evaluate XPath queries in linear time.

The algorithms in both parts of the paper use factorization forests.

This paper develops the use of factorization forests [8] for efficient evaluation of automata. The paper has two parts. The first part, which builds on [4,2], uses factorization forests to evaluate automata on arbitrary infixes of a word in constant time, after a linear time precomputation. The second part, which builds on [3,7], uses factorization forests to efficiently evaluate queries of XPath.

Infix evaluation. The first part of the paper studies the following problem, which is parametrized by a regular language $L \subseteq A^*$. For a word $a_1 \cdots a_n \in A^*$, we want to build a data structure. Then, we want to use the data structure to quickly answer queries of the form: given two positions $i \leq j$ in $\{1, \ldots, n\}$, answer if the infix $a_i \cdots a_j$ belongs to L. We call this the *infix evaluation problem*. A solution of the problem consists of two algorithms: the *preprocessing* that inputs $a_1 \cdots a_n$ and builds the data structure, and the *query answering* which inputs $i \leq j$ and outputs the answer to $a_i \cdots a_j \in L$.

A natural solution uses a divide and conquer approach. Suppose that L is recognized by a nondeterministic automaton with states Q. The preprocessing splits the word into halves, quarters, and so on. Each such infix is decorated with the set of state pairs that describe possible runs of the automaton over the infix. The preprocessing is in time $\text{poly}(Q) \cdot n$, while the query answering is in time $\text{poly}(Q) \cdot \log(n)$.

As observed by Thomas Colcombet in [4], a beautiful result of Imre Simon, called the Factorization Forest Theorem [8], can be used to answer the queries

* We acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under the FET-Open grant agreement FOX, number FP7-ICT-233599. Work supported by Polish government grant no. N N206 380037.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 515–526, 2010.

in time independent of the word's length. The data structure uses an algebraic approach to regular languages, where a language is recognized by a homomorphism from A^* into a finite monoid M. The preprocessing is in time linear in $|M| \cdot n$, and the query answering is in time linear in $|M|$.

What if the language L is given by an automaton and not a homomorphism? We can always compile the automaton to a monoid and use the above result. From the point of view of the length n of the word, the preprocessing is in linear time, and the query answering is in constant time. However, compiling even a deterministic automaton into a monoid can yield an exponential blowup. This gives big constants in the linear and constant times.

We can do better. If the language L is given by a deterministic automaton with states Q, a fairly straightforward structure, called the *tape construction* in [2], can be used to solve the problem with preprocessing in time $\text{poly}(Q) \cdot |n|$ and query answering in time $\text{poly}(Q)$.

In this paper, we improve the results from [4] and [2]: we give an algorithm that works with nondeterministic automata. As with the tape construction, the preprocessing is in time $\text{poly}(Q) \cdot n$ and the query answering in time $\text{poly}(Q)$. The new algorithm does not use the tape construction, which does not seem to generalize from deterministic to nondeterministic automata. Instead, it builds on factorization forests.

XPath evaluation. The second part of the paper is about XPath evaluation. The input for an XPath query is an XML document, which we model as a *data tree*. A data tree is a tree where each node carries two pieces of information: a tag name or label from a finite alphabet A, as well as a *data value* from an infinite alphabet D (such as integers, or unicode strings). An XPath query says "yes" or "no" to each node in a data tree. The XPath evaluation problem is to find the nodes to which the query says "yes".

There are algorithms which can solve this problem in time polynomial in the size of the query φ and the number of nodes n in the data tree, see [1] for a survey. However, with large XML documents (e.g. dblp.xml is currently 674 megabytes and millions of nodes), an algorithm that is quadratic in n is impractical. In previous work [3,7], we have developed algorithms which are linear in n.

The first algorithm, from [3], runs in time $\exp(\varphi) \cdot n$. The reason for the exponential complexity in the query is that parts of the query are represented by monoids. The algorithm works for an extension of XPath, called Regular XPath, which allows Kleene star in programs. The second algorithm, from [7], runs in time $\text{poly}(\varphi) \cdot n$. It works for XPath without the Kleene star. The general idea is that monoids can be avoided without the Kleene star. Both algorithms, especially the first one, use the ideas developed in the infix evaluation problem that is studied in the first part of this paper.

In the second part of this paper, we propose a new approach to XPath evaluation. We introduce an automaton model, which acts as an intermediate step between XPath and the evaluation algorithm. The automaton model is a type of transducer, which we call a *data aggregate transducer*. Given an input data tree, a data aggregate transducer produces new labels for the nodes, and does

not change the data values. A data aggregate transducer can evaluate a query by writing "yes" or "no" in the new label, depending on whether a node is selected. (Strictly speaking, we use compositions of data aggregate transducers to evaluate XPath queries.)

The advantage of this new approach is that the syntax of XPath is abstracted into a simple automaton model. This makes the evaluation algorithm easier to understand, and its structure more apparent. We also believe that the automata models we introduce, in the general form for data trees (data aggregate transducers), and in the restricted form for trees without data (which we call aggregate transducers), are of independent interest for evaluation algorithms.

The evaluation algorithm uses the algebraic techniques developed in the first part of the paper. Thanks to the efficient algorithms for nondeterministic automata (the path expressions in an XPath query are naturally modeled by nondeterministic automata), we get a new result: For data trees of bounded depth, a query φ of XPath with Kleene star can be evaluated in time $\text{poly}(\varphi) \cdot n$. In other words, for documents of bounded depth (a common situation), we can combine the efficient evaluation of [7] with the more powerful query language of [3].

1 Evaluating Infix Queries for Nondeterministic Automata

This section contains the first part of the paper, which talks about the infix evaluation problem. Instead of specifying an infix by its first x and last position y, we use the set of all of its positions $X = \{x, x+1, \ldots, y\}$. This way we can use set operations on infixes. If X is a set of positions in a word w, we write $w[X]$ for the subsequence of w consisting of positions from X, e.g. $a_1a_2a_3[\{1,3\}] = a_1a_3$. We use the name *factor* of w for a connected set of positions, and the name *infix* for the word $w[X]$ when X is a factor. (Of course, the algorithms represent factors by just keeping the first and last position.) We write x, y, z for positions, X, Y, Z for sets of positions, and F, G, H for factors (which are also sets of positions).

Factorization forests. A *factorization forest for a word* w is a family of factors that contains $\{x\}$ for every position x in w, and where every two factors are either disjoint, or one is contained in the other. There is a natural forest structure on the factors, so we can talk about descendants, parents, children and siblings, etc. The *level* of a factor is the number of its ancestors (including itself).

Suppose that $F_1, \ldots, F_n$ are consecutive factors (i.e. the first position of F_{i+1} is the next position after the last position of F_i). A *collation* of these factors is any union of these factors that is also a factor, i.e. any $F_i \cup \cdots \cup F_j$ for $i \leq j$. Consider a morphism $\alpha : A^* \to M$ into a finite monoid, which we use to map factors into M. We say that $F_1, \ldots, F_n$ are α-homogeneous if all of their collations have the same value under α. A factorization forest is called α-homogeneous if any choice of at least three consecutive siblings is α-homogeneous.

Suppose that w is a word with a factorization forest $\mathcal{F}$ that is α-homogeneous for a morphism $\alpha : A^* \to M$. Colcombet observed in [4] that for any factor I of w, not necessarily from $\mathcal{F}$, its image under α can be calculated in time linear in

the height of $\mathcal{F}$. Our work builds on this observation. As a first step, we show that the time can be even logarithmic in the height of $\mathcal{F}$.

Logarithmic querying. For the algorithms, we represent a factorization forest $\mathcal{F}$ as follows. Each factor $F \in \mathcal{F}$ is represented by a record with its first and last position, and its image under α. Each position x contains a pointer to the record of the factor $\{x\}$.

Each factor record stores a pointer to its parent factor record, but also to some other ancestors, as described below. Let n be a number from 0 to the logarithm of the height of the factorization forest. Consider a factor $F \in \mathcal{F}$. We create a pointer from the record of F to the record of the 2^n-parent of F, call it G. (The 2^n-parent is the ancestor 2^n levels above.) This pointer is called the *accelerating pointer of length* 2^n. It is decorated by two elements of M, which are the images under α of the two factors below.

- *left*(F, G): positions from G that are strictly before all positions from F.
- *right*(F, G): positions from G that are strictly after all positions from F.

The number of accelerating pointers, and the time required to compute them, is $|\mathcal{F}| \cdot \log h$, where h is the height of $\mathcal{F}$. From now on, we assume in our algorithms that factorization forests are equipped with accelerating pointers.

The benefit of accelerating pointers is that one can go from a factor to any of its ancestors by following a number of accelerating pointers that is logarithmic in the height of the forest. This observation, together with the original idea of using factorization forests homogeneous with a morphism to calculate images of infixes, gives the following result.

Lemma 1. *Let $\alpha : A^* \to M$ be a morphism, and let $\mathcal{F}$ be an α-homogeneous factorization forest for a word $w \in A^*$. Using the accelerating pointers, the image under α of any factor can be calculated in time logarithmic in the height of $\mathcal{F}$.*

Combining the above lemma with a divide and conquer approach, we get a solution for the infix evaluation problem that has querying in time $\log(\log(|w|))$. This is because a divide and conquer approach yields a factorization forest with binary branching, and such a forest is α-homogeneous for any α.

1.1 Monoid of Binary Relations

Let Q be any finite set. We write M_Q for the monoid of binary relations Q, where the monoid operation is relation composition. In this section, we study factorization forests that are α-homogeneous, for some $\alpha : A^* \to M_Q$. The size of the monoid M_Q is exponential in the size of Q. The main result of the first part of this paper is that we can build a factorization forest without worrying about this exponential blowup.

Theorem 1. *Consider a morphism $\alpha : A^* \to M_Q$. For any word $w \in A^*$ we can find, in time* $\mathrm{poly}(Q) \cdot |w|$*, an α-homogeneous factorization forest for w of height at most*[1] $\mathrm{poly}(M_Q)$.

[1] The height can be even linear in $|M_Q|$, but it requires more care in the proof.

We describe the proof of this theorem in Section 1.2.

Corollary 1. *Let $L \subseteq A^*$ be a language recognized by a nondeterministic automaton with states Q. The infix evaluation problem for a word $w \in A^*$ can be solved with precomputation* $\mathrm{poly}(Q) \cdot |w|$ *and query answering in time* $\mathrm{poly}(Q)$.

Proof. The nondeterministic automaton can be identified with a morphism $\alpha : A^* \to M_Q$, which maps a word w to the set of pairs (p, q) such that the automaton has a run from p to q over the word. Using the above theorem, we can compute a factorization forest in time $\mathrm{poly}(Q) \cdot |w|$. The height of the forest may be exponential in Q, since the height is bounded by M_Q. However, we can use the logarithm from Lemma 1 to query answer infix queries in time $\mathrm{poly}(Q)$. □

1.2 Proof of Theorem 1

For the rest of this section, we fix the monoid M_Q and the morphism α. We write r, s, t for the binary relations which are elements of M_Q, and $r \circ s$ for composition of binary relations, which is the monoid operation.

Green's relations. Let r, s, t, t_1, t_2 below be elements of M_Q.

- r is called a prefix of s, written $r \geq_{\mathcal{R}} s$, if there is some t with $r \circ t = s$.
- r is called a suffix of s, written $r \geq_{\mathcal{L}} s$, if there is some t with $t \circ r = s$.
- r is called an infix of s, written as $r \geq_{\mathcal{J}} s$, if there are t_1, t_2 with $t_1 \circ r \circ t_2 = s$.
- If r is both a prefix and a suffix of s, we write $r \geq_{\mathcal{H}} s$.

These relations are called Green's relations. It is easy to see that each of Green's relations is a pre-order: it is both transitive and reflexive. The relations are not necessarily antisymmetric and therefore it makes sense to consider their connected components. For instance, we say that r and s are $\mathcal{R}$-equivalent, written $r \sim_{\mathcal{R}} s$, if both $r \geq_{\mathcal{R}} s$ and $s \geq_{\mathcal{R}} r$. An equivalence class is called an $\mathcal{R}$-class. Likewise for $\mathcal{L}$, $\mathcal{J}$ and $\mathcal{H}$.

In the algorithm, we will need to perform operations on M_Q in time $\mathrm{poly}(Q)$. One such operation is calculating composition $r \circ s$, this is easy to do. A problem that we will have to work around is that we do not know how to test $\mathcal{J}$-equivalence in time $\mathrm{poly}(Q)$. However, we can do this in some special cases, as stated in the following lemma.

Lemma 2. *Given $r, s \in M_Q$, we can calculate the following in time* $\mathrm{poly}(Q)$*:*

$$r \circ s, \qquad r \circ s \stackrel{?}{\sim}_{\mathcal{J}} r, \qquad r \circ s \stackrel{?}{\sim}_{\mathcal{J}} s.$$

Proof strategy. We present the proof strategy for Theorem 1.
The definition of α-homogeneous factors or factorization forests also makes sense in a more general setting, where α is any function that maps factors of $\mathcal{F}$ to some set, not necessarily a morphism. We use this generalization to define notions of $\mathcal{J}$-homogeneity and $\mathcal{H}$-homogeneity. Let $F_1, \ldots, F_n$ be consecutive factors. We say the factors are $\mathcal{J}$-homogeneous if they are f-homogeneous under the function

f that maps a factor to the $\mathcal{J}$-class of its image. (In general, f is not a morphism.) Likewise we define a $\mathcal{J}$-homogeneous factorization forests, and the same for $\mathcal{H}$.

Our proof strategy is to first compute a $\mathcal{J}$-homogeneous factorization forest, then upgrade it to an $\mathcal{H}$-homogeneous one, and then upgrade that one to an α-homogeneous one. The main difficulty is in the first step – computing a $\mathcal{J}$-homogeneous forest; we do this below in Lemma 3. The other steps are done using basically the same techniques as in the proof of the factorization forest theorem from [6], or to the proofs of [8,4].

Lemma 3. *Let $w \in A^*$. One can compute a $\mathcal{J}$-homogeneous factorization forest $\mathcal{F}$ in time* $\mathrm{poly}(Q) \cdot |w|$. *The forest has height linear in M_Q.*

Proof. The algorithm processes word positions from left to right. We begin by describing the invariant.

The invariant. After processing position x, the algorithm will have computed a factorization forest $\mathcal{F}_x$ for the prefix $1, \ldots, x$. For each factor we remember one additional bit: if the factor is *open* or *closed*. All open factors have to contain the last processed position x. Open factors might grow when processing new positions. Once a factor becomes closed, it does not change. All singleton factors are closed. Suppose $F_1, \ldots, F_n \in \mathcal{F}_x$ is a maximal set of siblings (written from left to right). The invariant is that they satisfy the following property $\star$:

$\star$ The factors $F_1, \ldots F_{n-1}$, and the factor $F_1 \cup \cdots \cup F_{n-1}$ are all $\mathcal{J}$-equivalent.

Additionally, when they are children of an open factor $F \in \mathcal{F}_x$, the following property $\star\star$ is satisfied:

$\star\star$ F and F_1 are $\mathcal{J}$-equivalent.

The invariant is satisfied by the initial configuration $\mathcal{F}_1 = \{\{x\}\}$.

Once we have processed the whole word, it is not difficult to get a $\mathcal{J}$-homogeneous factorization forest from the one produced by the algorithm. For each maximal set of siblings $F_1, \ldots, F_n \in \mathcal{F}_x$, it is enough to add a factor $F_1 \cup \cdots \cup F_{n-1}$.

Updating the forest. Suppose we have computed $\mathcal{F}_{x-1}$, and we want to compute $\mathcal{F}_x$. Consider the factors open in $\mathcal{F}_{x-1}$:

$$x - 1 \in F_1 \subsetneq F_2 \subsetneq \cdots \subsetneq F_n.$$

There are also closed factors containing $x - 1$, at least one: $\{x - 1\}$. Let C be the biggest of them. We obtain $\mathcal{F}_x$ from $\mathcal{F}_{x-1}$ as follows.

- Add $\{x\}$.
- If C and F_1 are not $\mathcal{J}$-equivalent, or $n = 0$, add open factor $G_0 = C \cup \{x\}$.
- Replace the factors F_i by $G_i = F_i \cup \{x\}$, for $i \in \{1, \ldots, n\}$.
- When $G_i \backslash \{x\}$ and G_i are not $\mathcal{J}$-equivalent close G_i, for $i = 0$ (if G_0 was added) and for $i \in \{1, \ldots, n\}$.

The test on $\mathcal{J}$-equivalence in the second and the last step is done using Lemma 2, since we are testing $\mathcal{J}$-equivalence of a factor and its suffix or prefix. Below we argue that the invariant is preserved. Then, we show why the algorithm runs in the required time, and why the factorization forest has height linear in M_Q.

Correctness. Extending a factor does not impact on property $\star$, as it does not talk about a last sibling. Property $\star$ has to be checked only for the siblings of the newly added factor $\{x\}$. If G_0 is created, $\{x\}$ has only one sibling, so $\star$ is satisfied. Otherwise C is no longer the last sibling. This happens only when C is $\mathcal{J}$-equivalent to its parent F_1. As F_1 is open, it is $\mathcal{J}$-equivalent to its first child (from $\star\star$), hence to all its children (from $\star$), which gives $\star$ in the new forest.

Now check the property $\star\star$ for open factors. Factor G_0 stays open only when G_0 and $G_0 \backslash \{x\} = C$ are $\mathcal{J}$-equivalent, which is exactly $\star\star$. Any other G_i stays open when it is $\mathcal{J}$-equivalent to F_i, which (from $\star\star$) is equivalent to its first child (which is also the first child of G_i).

Running time. A potential problem is the last step. Potentially we have to do n tests for $\mathcal{J}$-equivalence. However notice that when $G_i \backslash \{x\}$ and G_i are $\mathcal{J}$-equivalent for some i, then they are $\mathcal{R}$-equivalent (Lemma **??**), hence also $G_j \backslash \{x\}$ and G_j are $\mathcal{R}$-equivalent ($\mathcal{J}$-equivalent) for any $j > i$. Thus we may stop testing greater i when we detect an equivalence. The number of tests for $\mathcal{J}$-equivalence is bounded by the number of factors becoming closed (plus one). Since the total number of factors in a factorization forest is at most twice the length of the word, we have a limit on the total number of operations in the last step of the algorithm.

Two implementation problems remain. First, where do we get the images of the factors $F_1, \ldots, F_n$ that are used in the tests for $\mathcal{J}$-equivalence? The answer is that our algorithm maintains for each open factor F_j, the image of its closed part $F_j - F_{j-1}$. Second, what is the cost of adding x to the factors F_i? The answer is that this can achieved for free, if we do not store the ends of open factors, but we only keep in mind that they all end in the currently processed position x.

Height of the forest. Why is the height of the factorization forest linear in M_Q? It would be useful to look at the $\mathcal{J}$-class of the first child of each non-singleton factor. The following invariant is preserved by the algorithm: whenever a factor F in the factorization forest is the parent of a non-singleton factor G, then the first child of F has a smaller $\mathcal{J}$-class than the first child of G. It guarantees that the level of a factor is bounded by the position of its first child in the $\leq_{\mathcal{J}}$ order.

Why is the invariant satisfied? First observe an auxiliary property of the forest: every closed factor in the factorization forest (except singletons) has a different (smaller) $\mathcal{J}$-class than its first child. Indeed, when a factor G_i becomes closed, it has a different $\mathcal{J}$-class than $G_i \backslash \{x\}$, which contains the first child of G_i.

To prove the invariant notice that during execution of the algorithm, the first child of a factor is never modified. Hence it is enough to analyze each moment when a new pair of a parent and its child is created. It happens only in the second step, when G_0 is created (creating $\{x\}$ does not matter, as the invariant does not

talk about singleton factors). First compare G_0 with its only non-singleton child C. As C is closed, from the above we know that its first child has greater $\mathcal{J}$-class than C itself, which is the first child of G_0. Now compare G_0 with its parent G_1. The factor G_0 is created only when C (the first child of G_0 has greater $\mathcal{J}$-class than F_1. Because F_1 is open, from $\star\star$ we get that it is $\mathcal{J}$-equivalent with its first child (which is also the first child of G_1). □

2 Aggregate Transducers

In this part of the paper, we introduce a new transducer model for data trees. This transducer is designed so that: a) it can compute interesting properties, such as XPath queries; b) it can be evaluated in linear time.

2.1 Trees without Data

Basic definitions. We work on finite, labeled, sibling-ordered trees. The trees are unranked, which means that there is no restriction on the number of children of a node. We use the usual notions of node, root, child, parent, descendant, ancestor etc. We write $t(x)$ for the label assigned by the tree t to the node x. We write trees(A) for the set of trees labeled by alphabet A. To recognize tree languages, we use nondeterministic automata on unranked trees. The exact choice of automaton model is not important for the discussion here; we choose nondeterministic finite hedge automata as defined in Section 8.2.2 of [5].

Transducers. Let A be an input alphabet and B an output alphabet. If s and t are trees with the same nodes, over alphabets A and B, then we write $s \otimes t$ for the tree over alphabet $A \times B$ that has the same nodes as s, t and maps each node x to the pair $(s(x), t(x))$. Consider a tree language over the product alphabet $A \times B$. This language can interpreted as a binary relation

$$f \subseteq \text{trees}(A) \times \text{trees}(B)$$

which contains a pair of trees (s, t) if the tree $s \otimes t$ belongs to the language. Note that the relation only contains tree pairs that have the same nodes. This type of relation is called a *transducer*. We use functional notation for transducers, writing $f(s)$ for the set of trees t with $(s, t) \in f$. We say a tree automaton *represents* f if it represents the underlying tree language over alphabet $A \times B$.

Aggregation. Consider an alphabet B equipped with a linear order. Suppose that s and t are trees over B that have the same nodes. We use the linear order to define a new tree, written $s \sqcup t$, which we call the *aggregation of s and t*. The tree $s \sqcup t$ has the same nodes as s and t, it assigns to a node x the bigger of the labels $s(x), t(x)$. The aggregation operation is commutative and associative, and therefore it makes sense to talk about the aggregation $\sqcup S$ of a set S of trees which share the same nodes.

Aggregate transducers. Suppose that f is a transducer with input alphabet A and output alphabet B. Suppose also that B is equipped with a linear order. Consider the function, call it $\sqcup f$, defined as

$$s \in \text{trees}(A) \qquad \mapsto \qquad \sqcup f(s) = \bigsqcup_{t \in f(s)} t \quad \in \text{trees}(B).$$

The notation $\sqcup f(s)$ is unambiguous, since $(\sqcup f)(s)$ and $\sqcup(f(s))$ mean the same thing. If $f(s)$ is empty, we define $\sqcup f(s)$ to be the tree with nodes from s labeled by the minimal element of B. Note that while f maps each tree to a set of trees, the function $\sqcup f$ maps each tree to a single tree. Any function of the form $\sqcup f$ is called an *aggregate transducer*. We believe that aggregate transducers are of independent interest.

2.2 Trees with Data

Data trees. Fix an infinite domain D of data values, e.g. $D = \mathbb{N}$. A *data tree* over a finite alphabet A is a tree over alphabet $A \times D$. The set of all data trees over an alphabet A is denoted $\text{dtrees}(A)$. We write such trees as $t \otimes \mu$, where t is a tree over A and μ a tree over D. The *label* of a node is its label in t, its *data value* is its label in μ. We use the name *class* for a set of nodes with the same data value. We assume that the data values are not greater than the number of nodes; thanks to this the classes can be found in time linear in the tree size. Data trees will be our document model for XPath queries[2].

Data aggregate transducer. We overload the $\otimes$ notation for sets as follows: if t is a tree over A and X is a set of nodes, we write $t \otimes X$ for the tree over $A \times \{0,1\}$, where the label of each node in t is enriched by a bit indicating membership in X. Consider a transducer f with input alphabet $A \times \{0,1\}$ and output alphabet B. Suppose also that B is equipped with a linear order so that trees over B can be aggregated. Consider the function, call it $\widehat{f}$, defined as

$$s \otimes \mu \in \text{dtrees}(A) \qquad \mapsto \qquad \widehat{f}(s \otimes \mu) = \bigsqcup_{X \text{a class of } \mu} \sqcup f(s \otimes X) \quad \in \text{trees}(B).$$

This is a function that maps a data tree over A to a tree without data over B. We use the name *data aggregate transducer* for any such function. An automaton *representing* $\widehat{f}$ is any automaton representing f. Note that since $\sqcup f(s)$ is itself an aggregation, the output $\widehat{f}(s \otimes \mu)$ is

$$\bigsqcup_{X \text{a class of } \mu} \quad \bigsqcup_{t \in f(s \otimes X)} t.$$

[2] In XML instead of small numbers we have arbitrary strings; however they can be sorted lexicographically and replaced by numbers in time linear in their total size. This is true even when the string value in an element node is not given explicitly, but is a concatenation of string values in its children, see [7].

The main motivation behind data aggregate transducers is that they can be used to evaluate XPath queries. We show this in Section 3.

Evaluation. The principal result on data aggregate transducers is that they can be evaluated in linear time. We have two variants of this result. The first variant works for the general case of data trees, but the constant in the linear time is exponential in the state space of the data aggregate transducer. The second variant has a polynomial constant, but it works only for data words, which are the special case of data trees where each node has at most one child.

Theorem 2. *Let $\widehat{f}$ be a data aggregate transducer represented by a nondeterministic tree automaton with states Q. The output of $\widehat{f}$ on a data tree $t \otimes \mu$ can be evaluated in time*

- $exp(Q) \cdot |t|$ *in the general tree case;*
- $\mathrm{poly}(Q) \cdot |t|$ *if $t \otimes \mu$ is a data word, i.e. each node has one child.*

We do not know if the variant for the general tree case can be improved to run in time $\mathrm{poly}(Q) \cdot |t|$.

2.3 Evaluating a Data Aggregate Transducer on Data Words

In this section, we prove the word case of Theorem 2, which says that data aggregate transducers can be evaluated in linear time. The tree case is done in the appendix. Instead of writing a data word as a tree where each node has one child, we use the standard notation for words as sequences of letters $a_1 \cdots a_n$.

We fix a data aggregate transducer $\widehat{f}$, and a nondeterministic automaton $\mathcal{A}$ of states Q that recognizes the underlying transducer f. The input alphabet of $\mathcal{A}$ is $A \times \{0,1\} \times B$. Fix also an input data word $w \otimes \mu$ of length n. We want to compute the output $\widehat{f}(w \otimes \mu)$. When talking about factors, we mean factors in a word of length n.

Snippets. We write $\bot$ for the minimal letter in the output alphabet B of f. For a word v over alphabet B and a set of positions Y, we we write $s \sqcap Y$ for the word obtained from v by replacing the labels of positions outside Y by $\bot$. A *partial output* is any value $(\sqcup f(w \otimes X)) \sqcap Y$ for some class X. A *snippet* is a partial output in which Y is a factor that is either disjoint with X, or included in X.

The *type* of a word $v \in (A \times \{0,1\})^*$ is the set of state pairs (p,q) such that $\mathcal{A}$ has a run from p to q over $v \otimes u$ for some $u \in B^*$. The *internal type* of a factor Y in a word $w \otimes X$ is the type of the corresponding infix. Let Y_1 (respectively, Y_2) consist of all positions before (after) a factor Y. The *external type* of the factor Y in a word $w \otimes X$ is the set of state pairs (p,q) such that (q_I, p) is in the internal type of Y_1 and (q, q_F) is in the internal type of Y_2 for an initial state q_I and an accepting state q_F. The external type of Y can be deduced from the internal types of Y_1 and Y_2.

We will use a concise representation for a snippet $(\sqcup f(w \otimes X)) \sqcap Y$. The *snippet representation* consists of: the factor Y, its external type in $w \otimes X$ and a membership bit saying whether Y is contained in X or disjoint with X. Note

that this information determines the value of $(\sqcup f(w \otimes X)) \sqcap Y$, as a word in B^*, even without knowing X.

We now have the necessary concepts to present our proof strategy for Theorem 2. Our goal is to produce the output $\widehat{f}(w \otimes \mu)$. Our algorithm will represent this output as the aggregation of a set of snippets. Whenever a subroutine of the algorithm inputs or outputs a set of snippets, we assume that the snippets are given by their representations.

The algorithm works in three stages.

Stage 1. We compute two factorization forests. Consider two words

$$w_0 = w \otimes \emptyset, \qquad w_1 = w \otimes \{1, \ldots, n\} \quad \in (A \times \{0,1\})^*.$$

We will use factorization forests for these words, for the morphism

$$\alpha : (A \times \{0,1\})^* \to M_Q$$

which maps a word v to its type. Apply Theorem 1 to the words w_0, w_1 and the morphism α, yielding factorization forests $\mathcal{F}_0$, $\mathcal{F}_1$. These factorization forests will be used by the next two stages of the algorithm.

Stage 2. We show that for each class X, the output $\sqcup f(w \otimes X)$ can be represented by a small number of snippets. This is stated by the following lemma.

Lemma 4. *Let X be a set of positions. We can calculate a set S_X of snippets such that $\sqcup f(w \otimes X) = \sqcup S_X$. The cardinality of S_X and time to calculate it are* $\mathrm{poly}(Q) \cdot |X|$.

Proof. Let $Y_1, \ldots, Y_m$ be a partition of $\{1, \ldots, n\}$ into factors such that the odd numbered factors are the maximal factors contained in X, and the even numbered ones are disjoint with X. (The first and last factors might be empty.) The set S_X consisting of the snippets $(\sqcup f(w \otimes X)) \sqcap Y_i$ satisfies the thesis. The factors Y_i can be calculated in time linear in the number of positions in X.

We need to find the representation of the snippets, namely the external types of Y_i in $w \otimes X$. The calculation will take time linear in m, and therefore at most linear in the size of X. Let first compute their internal types. For even i, the internal type of Y_i is $\alpha(w \otimes X[Y_i]) = \alpha(w_0[Y_i])$, hence we can compute it using the factorization forest $\mathcal{F}_0$. Using $\mathcal{F}_1$, we can do the same for odd i. Using compositionality of types, we calculate internal types of $Y_1 \cup \cdots \cup Y_i$ for each i, going from left to right, and of $Y_i \cup \cdots \cup Y_m$, going from right to left. The external type of Y_i is found basing on the internal types of $Y_1 \cup \cdots \cup Y_{i-1}$ and $Y_{i+1} \cup \cdots \cup Y_m$. □

Stage 3. In the third and final stage, we show that snippets can be efficiently aggregated. We apply Lemma 4 to each class X, yielding a set of snippets S_X. All we have to do is to aggregate them, i.e. aggregate all snippets that belong to some S_X for some class X. This can be done in linear time thanks to the following proposition.

Proposition 1. *Let $s_1, \ldots, s_m$ be snippets. Their aggregation $s_1 \sqcup \cdots \sqcup s_m$ can be calculated in time* $\mathrm{poly}(Q) \cdot (m + |w|)$.

3 An Application to Evaluating Queries of Regular XPath

Regular XPath is a logic for data trees, which extends XPath 1.0 by adding a Kleene star. There are two kinds of formulas in Regular XPath: unary queries and binary queries. A unary query maps a data tree to a set of nodes, and a binary query maps a data tree to a set of node pairs. The formulas of Regular XPath and their semantics, as we use them here, are defined in [3].

Theorem 3. *Let φ be a unary query of Regular XPath. The set of nodes selected by φ in a data tree with n nodes can be computed in time:*

- $exp(\varphi) \cdot n$; *or*
- $\text{poly}(\varphi) \cdot n$; *if the input is a word.*

The proof, given in the appendix, is straightforward: describe a query using data aggregate transducers, and apply Theorem 2. We would like to point out that a query is not described by a single data aggregate transducer, but a sequential composition, where each new transducer reads the output of the previous one.

Corollary 2. *Let φ be a unary query of Regular XPath. The nodes selected by φ in a data tree of height k with n nodes can be computed in time* $\text{poly}(k, \varphi) \cdot n$.

Proof. A data tree $t \otimes \mu$ of height k over an alphabet A can be encoded, by writing the nodes in document order and decorating them with their depths, as a data word $enc_k(t \otimes \mu)$ over alphabet $A \times \{1, \ldots, k\}$. This encoding can be decoded by Regular XPath in the following sense: for each unary query φ we can compute in time $\text{poly}(k, \varphi)$ a query $enc_k(\varphi)$ such that the set of nodes selected by φ in $t \otimes \mu$ can be recovered in linear time from the set of nodes selected by $enc_k(\varphi)$ in the data word $enc_k(t \otimes \mu)$. The idea is to replace the axes: e.g. next sibling is replaced by a disjunction, over all $i \in \{1, \ldots, k\}$, of the binary query which connects a position x of depth i with the first position $y > x$ such that y has depth i and all positions between x and y have depth at least $i + 1$. The Kleene star is needed to talk about the positions between x and y. □

References

1. Benedikt, M., Koch, C.: XPath leashed. ACM Comput. Surv. 41(1) (2008)
2. Bojanczyk, M.: Factorization forests. In: Developments in Language Theory, pp. 1–17 (2009)
3. Bojanczyk, M., Parys, P.: XPath evaluation in linear time. In: PODS, pp. 241–250 (2008)
4. Colcombet, T.: On factorisation forests. CoRR, abs/cs/0701113 (2007)
5. Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications (2007), http://www.grappa.univ-lille3.fr/tata (release October 12, 2007)
6. Kufleitner, M.: The height of factorization forests. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 443–454. Springer, Heidelberg (2008)
7. Parys, P.: XPath evaluation in linear time with polynomial combined complexity. In: PODS, pp. 55–64 (2009)
8. Simon, I.: Factorization forests of finite height. Theor. Comput. Sci. 72(1), 65–94 (1990)

On the Complexity of Searching in Trees: Average-Case Minimization*

Tobias Jacobs[1,**], Ferdinando Cicalese[2], Eduardo Laber[3], and Marco Molinaro[4]

[1] National Institute of Informatics, Japan
[2] University of Salerno, Italy
[3] PUC – Rio de Janeiro, Brazil
[4] Carnegie Mellon University, USA

Abstract. The well known binary search method can be described as the process of identifying some marked node from a line graph T by successively querying edges. An edge query e asks in which of the two subpaths induced by $T \setminus e$ the marked node lies. This procedure can be naturally generalized to the case where $T = (V, E)$ is a tree instead of a line. The problem of determining a tree search strategy minimizing the number of queries in the worst case is solvable in linear time [Onak *et al.* FOCS'06, Mozes *et al.* SODA'08].

Here we study the average-case problem, where the objective function is the weighted average number of queries to find a node An involved analysis shows that the problem is $\mathcal{NP}$-complete even for the class of trees with bounded diameter, or bounded degree.

We also show that any optimal strategy (i.e., one that minimizes the expected number of queries) performs at most $O(\Delta(T)(\log |V| + \log w(T)))$ queries in the worst case, where $w(T)$ is the sum of the node weights and $\Delta(T)$ is the maximum degree of T. This structural property is then combined with a non-trivial exponential time algorithm to provide an FPTAS for the bounded degree case.

1 Introduction

Searching is one of the fundamental problems in Computer Science and Discrete Mathematics. In his classical book [16], D. Knuth discusses many variants of the searching problem, most of them dealing with totally ordered sets. There has been some effort to extend the available techniques for searching and for other fundamental problems (e.g. sorting and selection) to handle more complex structures such as partially ordered sets [22,9,25,24,6]. Here, we focus on searching in structures that lie between totally ordered sets and the most general posets. We wish to efficiently locate a particular node in a tree.

* Most omitted proofs can be found at http://arxiv.org/abs/0904.3503.

** This work was supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD).

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 527–539, 2010.

More formally, as input we are given a tree $T = (V, E)$ which has a 'hidden' *marked* node and a function $w : V \to \mathbb{Z}^+$ that gives the likelihood of a node being the one marked. In order to discover which node of T is marked, we can perform *edge queries*: after querying the edge $e \in E$ we receive an answer stating in which of the two connected components of $T \setminus e$ the marked node lies. To simplify our notation let us assume that our input tree T is rooted at a node r. Then, we can specify a query to an edge $e = uv$, where u is the parent of v, by referring to v.

A search strategy is a procedure that decides the next query to be posed based on the outcome of the previous queries. Every search strategy for a tree $T = (V, E)$ (or for a forest) can be represented by a binary search (decision) tree D such that a path from the root of D to a leaf ℓ indicates which queries should be made at each step to discover that ℓ is the marked node. More precisely, a search tree for T is a triple $D = (N, E', A)$, where N and E' are the nodes and edges of a binary tree and the assignment $A : N \to V$ satisfies the following properties: (a) for every node v of V there is exactly one leaf ℓ in D such that $A(\ell) = v$; (b)[search property] if v is in the right (left) subtree of u in D then $A(v)$ is (not) in the subtree of T rooted at $A(u)$.

Given a search tree D for T, let $d(u, v)$ be the length (in number of edges) of the path from u to v in D. Then the cost of D, or alternatively the expected number of queries of D is given by

$$cost(D) = \sum_{v \in leaves(D)} d(root(D), v) w(A(v)).$$

Therefore, our problem can be stated as follows: given a rooted tree $T = (V, E)$ with $|V| = n$ and a function $w : V \to \mathbb{Z}^+$, the goal is to compute a minimum cost search tree for T.

The state of the art. The variant of our problem in which the goal is to minimize the number of edge queries in the worst case has been studied in several recent papers [3,25,24]. An optimal strategy can be found in linear time [24]. Also, the worst-case version of searching in trees has been independently considered as the problem of ranking the edges of a tree [8,7,20].

By contrast, little was known (prior to this paper) about the average-case minimization. The known results amount to the $O(\log n)$-approximation obtained by Kosaraju *et al.* [17], and Adler and Heeringa [1] for the much more general binary identification problem, and the constant factor approximation algorithm given by two of the authors in [18]. However, the complexity of the average-case minimization of the tree search problem has so far been unknown.

Our results. We significantly narrow the gap of knowledge in the complexity landscape of the tree search problem under different points of view. We prove that the problem is $\mathcal{NP}$-Complete even for the class of trees with diameter at most 4. This results in a complete characterization of the problem's complexity with respect to the parametrization in terms of the diameter. In fact, the problem is polynomially solvable for the class of trees of diameter at most 3.

We also show that the tree search problem under average minimization is $\mathcal{NP}$-Complete for trees of degree at most 16. This substantially improves upon

the state of the art, the only known result in this direction being an $O(n \log n)$ time solution [13,12] for the class of trees with maximum degree 2. The hardness results are obtained by fairly involved reductions from the Exact 3-Set Cover (X3C) with multiplicity 3 [11]. We match this hardness result with an FPTAS. In order to obtain the FPTAS, we first devise a non-trivial Dynamic Programming based algorithm that, roughly speaking, computes the best possible search tree, among the search trees with height at most H, in $O(n^2 2^H)$ time. Then, we show that every tree T admits a minimum cost search tree whose height is $O(\Delta \cdot (\log n + \log w(T)))$, where Δ is the maximum degree of T and $w(T)$ is the total weight of the nodes in T. This bound allows us to execute the DP algorithm with $H = c \cdot \Delta \cdot (\log n + \log w(T))$, for a suitable constant c, obtaining a pseudo-polynomial time algorithm for trees with bounded degree. By scaling the weights w in a fairly standard way we obtain the FPTAS.

We believe that the bound on the height of optimal search trees is of independent interest: it is nearly tight since the height of a search tree for a complete tree of degree Δ is $\Omega(\frac{\Delta}{\log \Delta} \log n)$. Moreover, it generalizes the known logarithmic bound on the height of optimal search trees for totally ordered sets [21].

In a paper which is in preparation, we show that the more general problem of computing an average-case optimal search strategy for partially ordered sets does not admit an $o(\log |V|)$-approximation unless $P = NP$. This sharply contrasts the existence of the FPTAS for bounded degree trees derived in this work.

Other related work. The problem studied here is a particular case of the binary identification problem (BIP) [10]: given is a set of elements $U = \{u_1, \ldots, u_n\}$, a set of tests $\{t_1, \ldots, t_m\}$, with $t_i \subseteq U$, a 'hidden' marked element, and a likelihood function $w : U \mapsto \mathbb{R}^+$. A test t determines whether or not the marked element is in the set t. The BIP asks to define a strategy that minimizes the (expected) number of tests to find the marked element. Both the average-case and the worst-case minimization are $\mathcal{NP}$-Complete [14], and neither admits an $o(\log n)$-approximation unless $\mathcal{P} = \mathcal{NP}$ [19,5]. For both versions, simple greedy algorithms attain $O(\log n)$-approximation [17,2,1]. Adding some structure to the set of tests yields interesting particular cases. If the set of tests is 2^U, then the optimal average cost strategy is a Huffman tree. Let G be a DAG with vertex set U. If the set of tests is $\{t_1, \ldots, t_n\}$, where $t_i = \{u_j | u_i \rightsquigarrow u_j \text{ in } G\}$, then we have the problem of searching in a poset [23,17,4]. When G is a directed path we have the alphabetic coding problem [13]. The problem we study here corresponds to the particular case where G is a directed tree.

2 Hardness

Our goal is to prove the following hardness results for the search tree problem.

Theorem 1. *The search tree problem is $\mathcal{NP}$-Complete in the class of trees of diameter at most 4. Moreover, it is also $\mathcal{NP}$-Complete in the class of trees of diameter at most 16.*

In this section we provide a proof for the first half of the theorem via reduction from the Exact 3-Set Cover problem with multiplicity bounded by 3, i.e., each element of the ground set can appear in at most 3 sets.

An instance of the 3-bounded Exact 3-Set Cover problem (X3C) is defined by: (a) a set $U = \{u_1, \ldots, u_n\}$, with $n = 3k$ for some $k \geq 1$; (b) a family $\mathcal{X} = \{X_1, \ldots, X_m\}$ of subsets of U, such that $|X_i| = 3$ for each $i = 1, \ldots m$ and for each $j = 1, \ldots n$, we have that u_j appears in at most 3 sets of $\mathcal{X}$. Given an instance $\mathbb{I} = (U, \mathcal{X})$ the X3C problem is to decide whether $\mathcal{X}$ contains a partition of U, i.e., whether there exists a family $\mathcal{C} \subseteq \mathcal{X}$ such that $|\mathcal{C}| = k$ and $\bigcup_{X \in \mathcal{C}} X = U$. This problem is well known to be $\mathcal{NP}$-Complete [11].

For our reduction it will be crucial to define an order among the sets of the family $\mathcal{X}$. Any total order $<$ on U, say $u_1 < u_2 < \cdots < u_n$, can be extended to a total order $\prec$ on $\mathcal{X} \cup U$ by stipulating that: (a) for any $X = \{x_1, x_2, x_3\}, Y = \{y_1, y_2, y_3\} \in \mathcal{X}$ (with $x_1 < x_2 < x_3$ and $y_1 < y_2 < y_3$,) the relation $X \prec Y$ holds if and only if the sequence $x_3\, x_2\, x_1$ is lexicographically smaller than $y_3\, y_2\, y_1$; (b) for every $j = 1, \ldots, n$, the relation $u_j \prec X$ holds if and only if the sequence $u_j\, u_1\, u_1$ is lexicographically smaller than $x_3\, x_2\, x_1$.

Assume an order $<$ on U has been fixed and $\prec$ is its extension to $U \cup \mathcal{X}$, as defined above. We denote by $\Pi = (\pi_1, \ldots, \pi_{n+m})$ the sequence of elements of $U \cup \mathcal{X}$ sorted in increasing order according to $\prec$. From now on, w.l.o.g., we assume that according to $<$ and $\prec$, it holds that $u_1 < \cdots < u_n$ and $X_1 \prec \cdots \prec X_m$. For each $i = 1, \ldots, m$, we shall denote the elements of X_i by $u_{i\,1}, u_{i\,2}, u_{i\,3}$ so that $u_{i\,1} < u_{i\,2} < u_{i\,3}$.

Example 1. Let $U = \{a, b, c, d, e, f\}$, $\mathcal{X} = \{\{a, b, c\}, \{b, c, d\}, \{d, e, f\}, \{b, e, f\}\}$. Fixing the standard alphabetical order among the elements of U, we have that the sets of $\mathcal{X}$ are ordered as follows: $X_1 = \{a, b, c\}$, $X_2 = \{b, c, d\}$, $X_3 = \{b, e, f\}$, $X_4 = \{d, e, f\}$. Then, we have $\Pi = (\pi_1, \ldots, \pi_{10}) = (a, b, c, X_1, d, X_2, e, f, X_3, X_4)$.

We shall first show a polynomial time reduction that maps any instance $\mathbb{I} = (U, \mathcal{X})$ of 3-bounded X3C to an instance $\mathbb{I}' = (T, w)$ of the tree search problem, such that T has diameter 4 but unbounded degree. We will then modify such reduction and show hardness for the bounded case too.

The structure of the tree T. The root of T is denoted by r. For each $i = 1, \ldots, m$ the set $X_i \in \mathcal{X}$ is mapped to a tree T_i of height 1, with root r_i and leaves $t_i, s_{i\,1}, s_{i\,2}, s_{i\,3}$. In particular, for $j = 1, 2, 3$, we say that $s_{i\,j}$ is associated with the element $u_{i\,j}$. We make each r_i a child of r. For $i = 1, \ldots, m$, we also create four leaves $a_{i1}, a_{i2}, a_{i3}, a_{i4}$ and make them children of the root r. We also define $\tilde{X}_i = \{t_i, s_{i\,1}, s_{i\,2}, s_{i\,3}, a_{i1}, \ldots, a_{i4}\}$ to be the set of leaves of T associated with X_i.

The weights of the nodes of T. Only the leaves of T will have non-zero weight, i.e., we set $w(r) = w(r_1) = \cdots = w(r_m) = 0$. It will be useful to assign weight also to each $u \in U$. In particular, our weight assignment will be such that each leaf in T which is associated with an element u will be assigned the same weight we assign to u. Also, when we fix the weight of u we shall understand that we are fixing the weight of all leaves in T associated with u. We extend the

function $w()$ to sets, so the weight of a set is the total weight of its elements. Also we define the weight of a tree as the total weight of its nodes.

The weights will be set in order to force any optimal search tree for (T, w) to have a well-defined structure. The following notions of Configuration and Realization will be useful to describe such a structure of an optimal search tree. In describing the search tree we shall use q_ν to denote the node in the search tree under consideration that represents the question about the node ν of the input tree T. Moreover, we shall in general only be concerned with the part of the search tree meant to identify the nodes of T of non-zero weight. It should be clear that the search tree can be easily completed by appending the remaining queries at the bottom.

Definition 1. *Given leaves $\ell_1, \ldots, \ell_h$ of T, a* sequential search tree for $\ell_1, \ldots, \ell_h$ *is a search tree of height h whose left path is $q_{\ell_1}, \ldots, q_{\ell_h}$. This is the strategy that asks about one leaf after another until they have all been considered.*

Configurations and realizations of Π. For each $i = 1, \ldots, m$, let D_i^A be the search tree with root q_{r_i}, with right subtree being the sequential search tree for $t_i, s_{i\,3}, s_{i\,2}, s_{i\,1}$, and left subtree being a sequential search tree for (some permutation of) $a_{i1}, \ldots a_{i4}$. We also refer to D_i^A as the A-configuration for $\tilde{X}_i$.

Moreover, let D_i^B be the search tree with root q_{t_i} and left subtree being a sequential search tree for (some permutation of) $a_{i1}, \ldots a_{i4}$. We say that D_i^B is the B-configuration for $\tilde{X}_i$.

Definition 2. *Given two search trees T_1, T_2, the extension of T_1 with T_2 is the search tree obtained by appending the root of T_2 to the leftmost leaf of T_1. The extension of T_1 with T_2 is a new search tree that "acts" like T_1 and in case of all NO answers continues following the strategy represented by T_2.*

Definition 3. *A realization (of Π) with respect to $\mathcal{Y} \subseteq \mathcal{X}$ is a search tree for (T, w) defined recursively as follows.*[1] *For each $i = 1, \ldots, n+m$, a realization of $\pi_i\, \pi_{i+1} \ldots \pi_{n+m}$ is an extension of the realization of $\pi_{i+1} \ldots \pi_{n+m}$ with another tree T' chosen according to the following two cases:*
Case 1. *If $\pi_i = u_j$, for some $j = 1, \ldots, n$, then T' is a (possibly empty) sequential search tree for the leaves of T that are associated with u_j and are not queried in the realization of $\pi_{i+1} \ldots, \pi_{n+m}$.*
Case 2. *If $\pi_i = X_j$, for some $j = 1, \ldots, m$, then T' is either D_j^B or D_j^A according to whether $X_j \in \mathcal{Y}$ or not.*

We denote by D^A the realization of Π w.r.t. the empty family, i.e., $\mathcal{Y} = \emptyset$.

We are going to set the weights in such a way that every optimal solution is a realization of Π w.r.t. some $\mathcal{Y} \subseteq \mathcal{X}$ (our Lemma 1). Moreover, such weights will allow to discriminate between the cost of solutions that are realizations w.r.t. an exact cover for the X3C instance and the cost of any other realization of Π. Let D^* be an optimal search tree and $\mathcal{Y}$ be such that D^* is a realization of Π w.r.t.

[1] For sake of definiteness we set $\pi_{m+n+1} = \emptyset$ and the realization of π_{n+m+1} w.r.t. $\mathcal{Y}$ to be the empty tree.

$\mathcal{Y}$.[2] In addition, for each $u \in U$ define $W_u = \sum_{\ell : X_\ell \prec u} w(\tilde{X}_\ell)$. It is not hard to see that the difference between the cost of D^A and D^*, can be expressed as

$$\sum_{X_i \in \mathcal{Y}} \left(w(t_i) - (W_{u_{i\,1}} + W_{u_{i\,2}} + W_{u_{i\,3}}) - \sum_{j=1}^{3} d_B^A(q_{s_{i\,j}}) w(u_{i\,j}) \right), \tag{1}$$

where $d_B^A(q_{s_{i\,j}})$ is the difference between the level of the node $q_{s_{i\,j}}$ in D^* and the level $q_{s_{i\,j}}$ in a realization of Π w.r.t. $\mathcal{Y} \setminus \{X_i\}$. To see this, imagine to turn D^A into D^* one step at a time. Each step being the changing of configuration from A to B for a set of leaves $\tilde{X}_i$ such that $X_i \in \mathcal{Y}$. Such a step implies: (a) moving the question $q_{s_{ij}}$ exactly $d_B^A(q_{s_{ij}})$ levels down, so increasing the cost by $d_B^A(q_{s_{i\,j}}) w(u_{i\,j})$; (b) because of (a) all the questions that were below the level where $q_{s_{ij}}$ is moved, are also moved down one level. This additional increase in cost is accounted for by the $W_{u_{i\,j}}$'s; (c) moving one level up the question about t_i, so gaining cost $w(t_i)$.

We will define the weight of t_i in order to: compensate the increase in cost (a)-(b) due to the relocation of $q_{s_{i\,j}}$; and to provide some additional gain only when $\mathcal{Y}$ is an exact cover. In general, the value of $d_B^A(q_{s_{i\,j}})$ depends on the structure of the realization for $\mathcal{Y} \setminus \{X_i\}$; in particular, on the length of the sequential search trees for the leaves associated to u_κ's, that appear in Π between X_i and $u_{i\,j}$. However, when $\mathcal{Y}$ is an exact cover, each such sequential search tree has length one. A moment's reflection shows that in this case $d_B^A(q_{s_{i\,j}}) = \gamma(i,j)$, where, for each $i = 1, \ldots, m$ and $j = 1, 2, 3$, we define

$$\gamma(i,j) = j - 5 + |\{u_\kappa : u_{i\,j} \prec u_\kappa \prec X_i\}| + 5 \cdot |\{X_\kappa : u_{i\,j} \prec X_\kappa \preceq X_i\}|$$

To see this, assume that $\mathcal{Y}$ is an exact cover. Let D' be the realization for $\mathcal{Y} \setminus X_i$, and ℓ be the level of the root of the A-configuration for $\tilde{X}_i$ in D'. The node $q_{s_{i\,j}}$ is at level $\ell + (5 - j)$ in D'. In D^*, the root of the B-configuration for $\tilde{X}_i$ is also at level ℓ. Also, in D^*, between level ℓ and the level of $q_{s_{i\,j}}$, there are only nodes associated with elements of some π_k s.t. $u_{ij} \prec \pi_\kappa \preceq X_j$. Precisely, there is 1 level per each u_κ s.t. $u_{ij} \prec u_\kappa \prec X_i$ (corresponding to the sequential search tree for the only leaf associated with u_κ); and 5 levels per each X_κ s.t. $u_{ij} \prec X_\kappa \preceq X_i$ (corresponding to the left path of the A or B-configuration for $\tilde{X}_\kappa$). In total, the difference between the levels of $q_{s_{ij}}$ in D' and D^* is $\gamma(i,j)$.

Note that $\gamma(i,j)$ is still well defined even if there is not an exact cover $\mathcal{Y} \subseteq \mathcal{X}$. This quantity will be used to define $w(t_i)$.

We are now ready to provide the precise definition of the weight function w. We start with $w(u_1) = 1$. Then, we fix the remaining weights inductively, using the sequence Π in the following way: let $i > 1$ and assume that for each $i' < i$ the weights of all leaves associated with $\pi_{i'}$ have been fixed[3]. We now proceed according to the following two cases:

[2] The existence of such a $\mathcal{Y}$ will be guaranteed by Lemma 1.

[3] By the leaves associated with $\pi_{i'}$ we mean the leaves in $\tilde{X}_j$, if $\pi_i = X_j$ for some $X_j \in \mathcal{X}$, or the leaves associated with u if $\pi_{i'} = u$ for some $u \in U$.

Case 1. $\pi_i = u_j$, for some $j \in \{1, \ldots, n\}$. Then, we set the weight of the nodes associated with u_j to $w(u_j) = 1 + 6\max\{|T|^3 w(u_{j-1}), W_{u_j}\}$, where $|T|$ denotes the number of nodes of T.
Case 2. $\pi_i = X_j$, for some $j \in \{1, \ldots, m\}$. Note that in this case the weights of the leaves $s_{j\,1}, s_{j\,2}, s_{j\,3}$ have already been fixed, respectively to $w(u_{j\,1}), w(u_{j\,2})$, and $w(u_{j\,3})$. This is because we fix the weights following the sequence Π and we have $u_{j\,1} \prec u_{j\,2} \prec u_{j\,3} \prec X_j$. In order to define the weights of the remaining elements in $\tilde{X}_j$ we set $w(a_{j1}) = \cdots = w(a_{j4}) = W_{u_{j\,1}} + W_{u_{j\,2}} + W_{u_{j\,3}} + \sum_{\kappa=1}^{3} \gamma(j,\kappa) w(u_{j\,\kappa})$. Finally, we set $w(t_j) = w(a_{j1}) + w(X_j)/2$.

Remark 1. For each $i = 1, \ldots, n+m$, let $w(\pi_i)$ denote the total weight of the leaves associated with π_i. It is not hard to see that $w(\pi_i) = O(|T|^{3i})$, So we have that the maximum weight is not larger than $w(\pi_{m+n}) = O(|T|^{3(m+n)})$. It follows that we can encode all weights using $O(3|T|(n+m)\log|T|)$ bits, hence the size of the instance (T, w) is polynomial in the size of the X3C instance $\mathbb{I} = (U, \mathcal{X})$.

Since t_m is the heaviest leaf, one can show that in an optimal search tree D^* the root can only be q_{t_m} or q_{r_m}. For otherwise moving one of these questions closer to the root of D^* results in a tree with smaller cost, violating the optimality of D^*. By a similar "exchange" argument it follows that if q_{r_m} is the root of D^* then the right subtree must coincide with a sequential search tree for $t_m, s_{m1}, s_{m2}, s_{m3}$ and the left subtree of q_{r_m} must be a sequential tree for $a_{m1}, \ldots, a_{m4}$. So the top levels of D^* coincide either with D^A_m or with D^B_m, or equivalently they are a realization of π_{m+n}. Repeating the same argument on the remaining part of D^* we have the following:

Lemma 1. *Any optimal search tree for the instance (T, w) is a realization of Π w.r.t. some $\mathcal{Y} \subseteq \mathcal{X}$.*

Recall now the definition of the search tree D^A. Let D^* be an optimal search tree for (T, w). Let $\mathcal{Y} \subseteq \mathcal{X}$ be such that D^* is a realization of Π w.r.t. $\mathcal{Y}$. Equation (1) and the definition of $w(t_i)$ yield

$$\begin{aligned} cost(D^A) - cost(D^*) &= \sum_{X_i \in \mathcal{Y}} \left(\frac{w(X_i)}{2} + \sum_{j=1}^{3} \left(\gamma(i,j) - d^A_B(q_{s_{i\,j}}) \right) w(u_{i\,j}) \right) \\ &= \sum_{j=1}^{n} \sum_{\substack{X_i \in \mathcal{Y} \\ u_j \in X_i}} \left(\frac{w(u_j)}{2} + \Gamma(i,j) w(u_j) \right), \end{aligned} \tag{2}$$

where $\Gamma(i,j) = \gamma(i,\kappa) - d^A_B(q_{s_{i\,\kappa}})$, and $\kappa \in \{1,2,3\}$ is such that $s_{i\,\kappa} = u_j$.

By definition, if for each $j = 1, \ldots, n$, there exists exactly one $X_i \in \mathcal{Y}$ such that $u_j \in X_i$, then we have $\Gamma(i,j) = 0$. Therefore, equation (2) evaluates exactly to $\sum_{j=1}^{n} \frac{w(u_j)}{2}$. Conversely, we can prove that this never happens when for some $1 \le j \le n$, u_j appears in none or in more than one of the sets in $\mathcal{Y}$. For this we use the exponential (in $|T|$) growth of the weights $w(u_j)$ and the fact that in such case the inner sum of the last expression in (2) is non-positive. In conclusion we have the following result.

Lemma 2. *Let D^* be an optimal search tree for (T, w). Let $\mathcal{Y} \subseteq \mathcal{X}$ be such that D^* is a realization of Π w.r.t. $\mathcal{Y}$. We have that $cost(D^*) \leq cost(D^A) - \frac{1}{2}\sum_{u \in U} w(u)$ if and only if $\mathcal{Y}$ is a solution for the X3C instance $\mathbb{I} = (U, \mathcal{X})$.*

The $\mathcal{NP}$-Completeness of 3-bounded X3C [11], Remark 1, and Lemma 2 imply the first half of Theorem 1. The hardness for trees of bounded diameter can be obtained by refining these hardness instances, carefully introducing nodes to control the degree expansion.

3 An FPTAS for Searching in Bounded-Degree Trees

First we need to introduce some notation. For any forest F of rooted trees and node $j \in F$, we denote by F_j the subtree of F composed by j and all of its descendants. We denote the root of a tree T by $r(T)$, $\delta(u)$ denotes the number of children of u and $c_i(u)$ is used to denote the ith child of u according to some arbitrarily fixed order. Given a search tree D for T, we use l_u to denote the leaf of D assigned to node u of T.

The following operation will be useful for modifying search trees: Given a search tree D and a node $u \in D$, a *left deletion* of u is the operation that transforms D into a new search tree by removing both u and its left subtree from D and, then, by connecting the right subtree of u to the parent of u (if it exists). A *right deletion* is analogously defined.

A dynamic programming algorithm. We often construct a search tree starting with its 'left part'. In order to formally describe such constructions, we define a *left path* as an ordered path where every node has only a left child. In addition, the *left path* of an ordered tree T is defined as the ordered path we obtain when we traverse T by only going to the left child, until we reach a node which does not have a left child.

In order to find an optimal search tree in an efficient way, we will define a family of auxiliary problems denoted by $\mathcal{P}^B(F, P)$. For this we first need to introduce the concept of an *extended search tree*, which is basically a search tree with some extra nodes that either have not been associated with a query yet (unassigned nodes), or cannot be associated with a query (blocked nodes).

Definition 4. *An* extended search tree *(EST) for a forest $F = (V, E)$ is a triple $D = (N, E', A)$, where N and E' are the nodes and edges of an ordered binary tree and the assignment $A : N \to V \cup \{blocked, unassigned\}$ simultaneously satisfy the following properties:*

(a) For every node v of F, D contains both a leaf ℓ and an internal node u such that $A(\ell) = A(u) = v$;
(b) $\forall u, v \in D$, with $A(u), A(v) \in F$, the following holds: If v is in the right subtree of u then $A(v) \in F_{A(u)}$. If v is in left subtree of u then $A(v) \notin F_{A(u)}$;
(c) If u is a node in D with $A(u) \in \{blocked, unassigned\}$, then u does not have a right child.

If we drop (c) and also the requirement regarding internal nodes in (a) we have the definition of a search tree for F. The cost of an EST D for F is analogous to the cost of a search tree and is given by $cost(D) = \sum d(r(D), u)w(A(u))$, where the summation is taken over all leaves $u \in D$ for which $A(u) \in F$.

At this point we establish a correspondence between optimal EST's and optimal search trees. Given an EST D for a tree T, we can apply a left deletion to the internal node of D assigned to $r(T)$ and right deletions to all nodes of D that are blocked or unassigned, getting a search tree D' of cost $cost(D') \leq cost(D) - w(r(T))$. Conversely, we can add a node assigned to $r(T)$ to a search tree D' and get an EST D such that $cost(D) \leq cost(D') + w(r(T))$. Employing these observations we can prove the following lemma:

Lemma 3. *Any optimal EST for a tree T can be converted into an optimal search tree for T (in linear time). In addition, the existence of an optimal search tree of height h implies the existence of an optimal EST of height $h + 1$.*

So we can focus on obtaining optimal EST's. First, we introduce concepts which serve as building blocks for EST's. A *partial left path* (PLP) is a left path where every node is assigned (via a function A) to either *blocked* or *unassigned*. Let D be an EST D and $L = \{l_1, \ldots, l_{|L|}\}$ be its left path. We say that D is *compatible* with a PLP $P = \{p_1, \ldots, p_{|P|}\}$ if $|P| = |L|$ and $A(p_i) = blocked$ implies $A(l_i) = blocked$.

This definition of compatibility implies a natural one to one correspondence between nodes of L and P. Therefore, without ambiguity, we can use p_i when referring to node l_i and vice versa.

Now we can introduce our subproblem $\mathcal{P}^B$. First, fix a tree T with n nodes and a weight function w. Given a forest $F = \{T_{c_1(u)}, T_{c_2(u)} \ldots, T_{c_f(u)}\}$, a PLP P and an integer B, the problem $\mathcal{P}^B(F, P)$ consists of finding an EST for F with minimum cost among the EST's for F compatible with P and have height at most B. Note that F is not a general subforest of T, but one consisting of subtrees rooted at the first f children of some node $u \in T$, for some $1 \leq f \leq \delta(u)$.

Notice that if P is a PLP where all nodes are unassigned and P and B are sufficiently large, then $\mathcal{P}^B(T, P)$ gives an optimal EST for T.

Algorithm for $\mathcal{P}^B(F, P)$. We have a base case and two other cases depending on the structure of F. In all cases, although not explicitly stated, the algorithm returns 'infeasible' if P contains no unassigned nodes. Whenever the algorithm encounters an 'infeasible' subproblem it ignores this choice in the enumeration.

Base case: *F has only one node u.* In this case, the optimal solution for $\mathcal{P}^B(F, P)$ is obtained from P by assigning its first unassigned node, say p_i, to u and then adding a leaf assigned to u as a right child of p_i. Its cost is $i \cdot w(u)$.

Case 1: *F is a forest $\{T_{c_1(u)}, \ldots, T_{c_f(u)}\}$.* The idea of the algorithm is to decompose the problem into subproblems for the forests $T_{c_f(u)}$ and $F \setminus T_{c_f(u)}$. For that, it needs to select which nodes of P will be assigned to each of these forests.

The algorithm considers all possible bipartitions of the unassigned nodes of P and for each bipartition $\mathcal{U} = (U^f, U^o)$ it computes an EST $D^{\mathcal{U}}$ for F compatible with P. At the end, the algorithm returns the tree $D^{\mathcal{U}}$ with smallest cost. The EST $D^{\mathcal{U}}$ is constructed as follows:

1. Let P^f be the PLP constructed by starting with P and then setting all nodes in U^o as blocked. Similarly, let P^o be the PLP constructed by starting with P and setting all nodes in U^f as blocked. Let D^f and D^o be optimal solutions for $\mathcal{P}^B(T_{c_f(u)}, P^f)$ and $\mathcal{P}^B(F \setminus T_{c_f(u)}, P^o)$, respectively.
2. The EST $D^{\mathcal{U}}$ is computed by taking the 'union' of D^f and D^o. More formally, the 'union' operation consists of starting with the path P and then replacing: (i) every node in $P \cap U^f$ by the corresponding node in the left path of D^f and its right subtree; (ii) every node in $P \cap U^o$ by the corresponding node in the left path of D^o and its right subtree.

Notice that the height of every EST $D^{\mathcal{U}}$ is at most B; this implies that the algorithm returns a feasible solution for $\mathcal{P}^B(F, P)$. Also, the cost of $D^{\mathcal{U}}$ is given by $OPT(\mathcal{P}^B(T_{c_f(u)}, P^f)) + OPT(\mathcal{P}^B(F \setminus T_{c_f(u)}, P^o))$.

The optimality of the above procedure relies on the fact we can build an EST $\bar{D}^f$ for $T_{c_f(u)}$ by starting from an optimal solution D^* for $\mathcal{P}^B(F, P)$ and performing the following operation at each node v of its left path: (i) if v is unassigned we assign it as blocked; (ii) if v is assigned to a node in $F \setminus T_{c_f(u)}$ we assign it as blocked and remove its right subtree. We can construct an EST $\bar{D}^o$ for $F \setminus T_{c_f(u)}$ analogously. Notice that $cost(\bar{D}^f) + cost(\bar{D}^o) = cost(D^*)$. The proof is then completed by noticing that, for a particular choice of $\mathcal{U}$, $\bar{D}^f$ and $\bar{D}^o$ are feasible for $\mathcal{P}^B(T_{c_f(u)}, P^f)$ and $\mathcal{P}^B(F \setminus T_{c_f(u)}, P^o)$, so the solution returned by the above algorithm costs at most $OPT(\mathcal{P}^B(T_{c_f(u)}, P^f)) + OPT(\mathcal{P}^B(F \setminus T_{c_f(u)}, P^o)) \leq cost(D^*)$.

Case 2: *F is a tree T_v.* Let p_i be an unassigned node of P and let t be an integer in the interval $[i+1, B]$. The algorithm considers all possibilities for p_i and t and computes an EST $D^{i,t}$ for T_v of smallest cost satisfying the following: (i) $D^{i,t}$ is compatible with P; (ii) its height is at most B; (iii) the node of the left path of $D^{i,t}$ corresponding to p_i is assigned to v; (iv) the leaf of $D^{i,t}$ assigned to v is located at level t. The algorithm then returns the tree $D^{i,t}$ with minimum cost.

In order to compute $D^{i,t}$ the algorithm executes the following steps:

1. Let P^i be the subpath of P that starts at the first node of P and ends at p_i. Let $P^{i,t}$ be a left path obtained by appending $t - i$ unassigned nodes to P^i and assigning p_i as blocked. Compute an optimal solution D' for $\mathcal{P}^B(\{T_{c_1(v)}, T_{c_2(v)}, \ldots, T_{c_{\delta(v)}(v)}\}, P^{i,t})$.
2. Let p'_i be the node of D' corresponding to p_i and let y' be the last node of the left path of D'. The tree $D^{i,t}$ is constructed by modifying D' as follows: make the left subtree of p'_i becomes its right subtree; assign p'_i to v; add a leaf assigned to v as the left child of y'; finally, as a technical detail, add

some blocked nodes to extend the left path of this structure until the left path has the same size of P.

It follows from properties (i) and (ii) of the trees $D^{i,t}$'s that the above procedure returns a feasible solution for $\mathcal{P}^B(T_v, P)$. The proof of the optimality uses the same type of arguments as in Case 1 and is deferred to the full version of the paper.

Computational complexity. Notice that it suffices to consider the problems $\mathcal{P}^B(F, P)$'s where $|P| \leq B$, since all others are infeasible. All these problems can be solved in $O(n^2 2^{2B})$ time, since there are $O(n2^B)$ such problems and each can be solved in $O(n + 2^B)$ time by employing a dynamic programming strategy.

From the DP algorithm to an FPTAS. The last piece that we need in order to obtain the FPTAS is a bound on the height of an optimal tree.

Theorem 2. *There is an optimal search tree for (T, w) of height at most $O(\Delta(T) \cdot (\log w(T) + \log n))$.*

Proof (sketch). First we notice the following:
Claim 1. For any subtree T' of T, we can obtain from D^* a search tree $D^*_{T'}$ for T', such that $d(r(D^*_{T'}), l_x) = d(r(D^*), l_x) - n_x$, where n_x is the number of nodes in the path from $r(D^*)$ to l_x assigned to nodes in $T - T'$.
Claim 2. Let D^* be an optimal search tree. Fix $0 \leq \alpha < 1$ and an integer $c > 3(\Delta(T) + 1)\alpha$. Then, for every node $v^* \in D^*$ with $d(r(D^*), v^*) \geq c$ we have that $w(D^*_{v^*}) \leq \alpha \cdot w(D^*)$.

Assume (by contradiction) Claim 2 does not hold for some v^* satisfying its conditions. Let $\tilde{T}$ be the tree associated with v^*, rooted at node $\tilde{r}$. Since by hypothesis $w(\tilde{T}) > \alpha \cdot w(D^*)$, we create the following search tree D' which makes sure parts of $\tilde{T}$ are queried closer to $r(D')$: the root of D' is assigned to $\tilde{r}$; the left tree of $r(D')$ is a search tree for $T - T_{\tilde{r}}$ given by Claim 1; in the right tree of $r(D')$ we build a left path containing nodes corresponding to queries for $c_1(\tilde{r}), c_2(\tilde{r}), \ldots, c_{\delta(\tilde{r})}(\tilde{r})$, each having as right subtree a search tree for the corresponding $T_{c_i(\tilde{r})}$ given by Claim 1. If $\bar{s}$ is the number of nodes of $T - T_{\tilde{r}}$ queried in $r(D^*) \rightsquigarrow v^*$, then Claim 1 implies that D' saves at least $\bar{s} - (\Delta(T) + 1)$ queries for each node in $\tilde{T}$ when compared to D^*; this gives the expression $cost(D') \leq cost(D^*) - \bar{s} \cdot w(\tilde{T}) + (\Delta(T)+1)w(T)$. Using the hypothesis on c and $w(\tilde{T})$, this is enough to reach the contradiction $cost(D') < cost(D^*)$ when $\bar{s} \geq c/3$. The case when $\bar{s} < c/3$ is a little more involved but uses a similar construction, only now the role of $\tilde{r}$ is taken by a node inside $T_{\tilde{r}}$ in order to obtain a more 'balanced' search tree.

Assume that the weight function w is strictly positive (the general case is in Appendix). Since w is integral, repeated use of Claim 2 yields the desired result.

Employing the previous dynamic programming algorithm with $B = O(\Delta(T) \cdot (\log w(T) + \log n))$, we obtain an optimal solution in time $O\left((n \cdot w(T))^{O(\Delta(T))}\right)$. Finally, by a standard weight scaling method (e.g. [15]), we obtain an FPTAS.

Theorem 3. *Consider an instance (T, w) to our search problem where $\Delta(T) = O(1)$. Then there is a poly$(n \cdot w(T))$-time algorithm for computing an optimal search tree for (T, w). In addition, there is a poly(n/ϵ)-time algorithm for computing an $(1 + \epsilon)$-approximate search tree for (T, w).*

References

1. Adler, M., Heeringa, B.: Approximating optimal binary decision trees. In: APPROX-RANDOM, pp. 1–9 (2008)
2. Arkin, E., Meijer, H., Mitchell, J., Rappaport, D., Skiena, S.: Decision trees for geometric models. Int. Jour. of Comp. Geom. and Appl. 8(3), 343–364 (1998)
3. Ben-Asher, Y., Farchi, E., Newman, I.: Optimal search in trees. SIAM Journal on Computing 28(6), 2090–2102 (1999)
4. Carmo, R., Donadelli, J., Kohayakawa, Y., Laber, E.: Searching in random partially ordered sets. Theoretical Computer Science 321(1), 41–57 (2004)
5. Chakaravarthy, V., Pandit, V., Roy, S., Awasthi, P., Mohania, M.: Decision trees for entity identification: Approximation algorithms and hardness results. In: PODS, pp. 53–62 (2007)
6. Daskalakis, C., Karp, R., Mossel, E., Riesenfeld, S., Verbin, E.: Sorting and selection in posets. In: SODA, pp. 392–401 (2009)
7. de la Torre, P., Greenlaw, R., Schäffer, A.: Optimal edge ranking of trees in polynomial time. Algorithmica 13(6), 592–618 (1995)
8. Dereniowski, D.: Edge ranking and searching in partial orders. DAM 156, 2493–2500 (2008)
9. Faigle, U., Lovász, L., Schrader, R., Turán, G.: Searching in trees, series-parallel and interval orders. SICOMP: SIAM Journal on Computing 15 (1986)
10. Garey, M.: Optimal binary identification procedures. SIAP 23(2), 173–186 (1972)
11. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP–Completeness. Freeman, New York (1979)
12. Garsia, A., Wachs, M.: A new algorithm for minimum cost binary trees. SIAM Journal on Computing 6(4), 622–642 (1977)
13. Hu, T., Tucker, A.: Optimal computer search trees and variable-length alphabetic codes. SIAM Journal on Applied Mathematics 21(4) (1971)
14. Hyafil, L., Rivest, R.: Constructing optimal binary decision trees is NP-complete. Information Processing Letters 5(1), 15–17 (1976)
15. Ibarra, O., Kim, C.: Fast approximation algorithms for the knapsack and sum of subset problems. Journal of the ACM 22(4), 463–468 (1975)
16. Knuth, D.: The Art of Computer Programming, Vol. 3: Sorting and Searching. Addison-Wesley, Reading (1973)
17. Kosaraju, R., Przytycka, T., Borgstrom, R.: On an optimal split tree problem. In: Dehne, F., Gupta, A., Sack, J.-R., Tamassia, R. (eds.) WADS 1999. LNCS, vol. 1663, pp. 157–168. Springer, Heidelberg (1999)
18. Laber, E., Molinaro, M.: An approximation algorithm for binary searching in trees. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 459–471. Springer, Heidelberg (2008)

19. Laber, E., Nogueira, L.: On the hardness of the minimum height decision tree problem. Discrete Applied Mathematics 144(1-2), 209–212 (2004)
20. Lam, T., Yue, F.: Optimal edge ranking of trees in linear time. In: SODA, pp. 436–445 (1998)
21. Larmore, H., Hirschberg, D.S., Larmore, L.L., Molodowitch, M.: Subtree weight ratios for optimal binary search trees. Technical report (1986)
22. Linial, N., Saks, M.: Searching ordered structures. J. of Algorithms 6 (1985)
23. Lipman, M., Abrahams, J.: Minimum average cost testing for partially ordered components. IEEE Transactions on Information Theory 41(1), 287–291 (1995)
24. Mozes, S., Onak, K., Weimann, O.: Finding an optimal tree searching strategy in linear time. In: SODA, pp. 1096–1105 (2008)
25. Onak, K., Parys, P.: Generalization of binary search: Searching in trees and forest-like partial orders. In: FOCS, pp. 379–388 (2006)
26. Schäffer, A.: Optimal node ranking of trees in linear time. IPL 33, 91–96 (1989)

Finding Is as Easy as Detecting for Quantum Walks

Hari Krovi[1], Frédéric Magniez[2], Maris Ozols[3,4], and Jérémie Roland[3]

[1] University of Connecticut
[2] LRI, Univ Paris-Sud, CNRS
[3] NEC Laboratories America, Inc.
[4] University of Waterloo and Institute for Quantum Computing

Abstract. We solve an open problem by constructing quantum walks that not only detect but also find marked vertices in a graph. The number of steps of the quantum walk is quadratically smaller than the classical hitting time of any reversible random walk P on the graph.

Our approach is new, simpler and more general than previous ones. We introduce a notion of interpolation between the walk P and the absorbing walk P', whose marked states are absorbing. Then our quantum walk is simply the quantum analogue of the interpolation. Contrary to previous approaches, our results remain valid when the random walk P is not state-transitive, and in the presence of multiple marked vertices.

As a consequence we make a progress on an open problem related to the spatial search on the 2D-grid.

1 Introduction

Many classical randomized algorithms rely heavily on random walks or Markov chains. The notion of hitting time is intimately related to the problem of spatial search, where displacement constraints are modeled by an undirected graph G. The set of desired vertices, or marked vertices, is denoted by M. Classically, a simple algorithm to find a marked vertex is to repeatedly apply some random walk P on G until one of the marked vertices is reached. The hitting time of P, $\mathrm{HT}(P, M)$, is precisely the expected number of repetitions, or steps, necessary to reach a marked vertex, starting from the stationary distribution of P.

Quantum walks are natural generalizations of classical random walks. Ambainis [1] was the first to solve a natural problem – the "element distinctness problem" – using a quantum walk. Following this, many quantum walk algorithms were discovered [2,3,4]. Quantum walk algorithms for the spatial search problem [5] were studied for the hypercube [6] and the grid [7,8].

The notion of hitting time has been carried over to the quantum case in [8,9,10,11,12,13]. Usually, the quantum hitting time has a quadratic improvement over the classical one. However, until the present paper, several serious restrictions were imposed: a quantum algorithm could only solve the detection problem of deciding whether there are marked vertices or not [10], but for being able to find them the Markov chain had to be reversible, state-transitive, and with

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 540–551, 2010.

a unique marked vertex [14,13]. The detection algorithm is quite intuitive and well understood, whereas the finding algorithm requires an elaborate proof whose intuition is not clear. This is due in part to a modification of the quantum walk, so that the resulting walk is not a quantum analogue of a Markov chain anymore.

Whether this quadratic speed-up for finding a marked vertex also holds for any reversible Markov chain and for multiple marked vertices was an open question. In this paper, we answer this question in the positive. Here we choose another approach by modifying directly P, and by considering the quantum analogue of the modified random walk. Doing that we keep some intuition and get simpler proofs while obtaining more general results. The new walk is simply an interpolation between the walk P and the absorbing walk P', where all outgoing transitions from marked vertices are replaced by self-loops. The interpolation coefficient can be used to tune the overlap between the stationary superposition of the quantum walk and its projection onto marked vertices. For a suitable value, depending on the relative weight of marked vertices in the stationary distribution of P, the overlap with marked and unmarked vertices is balanced, leading to a quantum walk algorithm that finds a marked vertex within $\sqrt{\mathrm{HT}(P, M)}$ steps (**Theorem 5**). The balancing can also be achieved when limited or even no information is available on the weight of marked vertices (**Theorems 6, 7 and 8**). As a consequence, we make a progress on an open problem from [5,14] related to the spatial search on the 2D-grid (**Corollary 2**).

2 Preliminaries

2.1 Spatial Search on Graphs

We fix throughout the paper an undirected graph $G = (X, E)$ with $|X| = n$. Let $M \subseteq X$ be a set of marked vertices and let $m = |M|$. Vertices are encoded in a distinguished *vertex register*. Our goal is to find any of the marked vertices in M using only evolutions that preserve the locality of G on the vertex register, i.e., to perform a *spatial search* on G [5]. Here we define an even more restricted notion of locality than the ones in [5], but it is more intuitive and sufficiently powerful for our purpose. We allow two types of operations on the vertex register: *static* transformations (that can be conditioned on the state of the vertex register but do not modify it) and shift (that exchanges the value of the vertex register and another register). Nonetheless, we want to restrict the executions of shift to $(x, y) \in E$.

Definition 1. *Let*

$$\textsc{Shift}\ (x, y) = \begin{cases} (y, x), & \text{if } (x, y) \in E, \\ (x, y), & \text{otherwise.} \end{cases}$$

In the first case we say that Shift succeeds*, but in the second case it* fails*.*

Definition 2 (Problems). *Under the restrictions that only static transformations and* Shift *are allowed, consider the following problems:*

- Detect(G)*: Detect if there is a marked vertex in* G*;*
- Find(G)*: Find any marked vertex in* G*, with the promise that* $M \neq \emptyset$*.*

$\textsc{Detect}^{(k)}(G)$ (resp. $\textsc{Detect}^{(\geq k)}(G)$) will denote the problem $\textsc{Detect}(G)$ with the promise that $m = 0$ or $m = k$ (resp. $m \geq k$). Similarly, define $\textsc{Find}^{(k)}(G)$ (resp. $\textsc{Find}^{(\geq k)}(G)$) as $\textsc{Find}(G)$ with the promise that $m = k$ (resp. $m \geq k$).

A natural approach to searching on a graph consists in using a random walk. Intuitively, a random walk is an alternation of coin flips and shifts. Namely, a coin is flipped according to the current state x of the vertex register, its value describes a target vertex y, and $\textsc{Shift}$ is a move from x to y. Let p_{xy} be the probability that x is shifted to y. Then $\textsc{Shift}$ always succeeds if $p_{xy} = 0$ whenever $(x, y) \notin E$. In that case, we say that $P = (p_{xy})_{x,y \in X}$ is a Markov chain on G.

We assume from now on that P is an ergodic Markov chain. Therefore P has a unique stationary distribution π. We also assume that P is reversible: $\pi_x p_{xy} = \pi_y p_{yx}$, for all $x, y \in X$. To measure the complexity of implementing a random walk corresponding to P, we introduce the following black-box operations:

- $\texttt{Check}(M)$: Check if a given vertex is marked;
- $\texttt{Setup}(P)$: Draw a sample from the stationary distribution π of P;
- $\texttt{Update}(P)$: Perform one step of P.

Each of these black-box operations have the corresponding associated implementation cost. We denote by C, S and U the respective complexities of the transformations $\texttt{Check}(M)$, $\texttt{Setup}(P)$ and $\texttt{Update}(P)$.

2.2 Quantum Version

In the quantum case, the problem extends as follows. Let $\mathcal{H} = \mathbb{C}^X$ be a fixed Hilbert space with basis $(|x\rangle)_{x \in X}$. Again, a transformation is *static* if it is controlled by the vertex register, that is of type $\sum_{x \in X} |x\rangle\langle x| \otimes V_x$, and Definition 1 of $\textsc{Shift}$ is simply extended by linearity. Then the generalization of random walks to quantum walks is as follows.

Definition 3. *A* quantum walk W *on* G *is a composition of static unitary transformations and* $\textsc{Shift}$ *on an invariant subspace of* $\mathcal{H} \otimes \mathcal{H}$*, the* walk space*, such that* $\textsc{Shift}$ *always succeeds when* W *is restricted to its walk space.*

Implicitly we always restrict a quantum walk to its walk space.

We will only consider quantum walks built from quantum analogues of reversible Markov chains. Thus we extend the operations `Check`, `Setup` and `Update` to the quantum setting as follows. Let $|\overline{0}\rangle \in \mathcal{H}$ be a fixed reference state. In the following, the first register is the vertex register.

- $\texttt{Check}(M)$: Map $|x\rangle|b\rangle$ to $|x\rangle|b\rangle$ if $x \notin M$ and $|x\rangle|b \oplus 1\rangle$ if $x \in M$, for $b = 0, 1$.
- $\texttt{Setup}(P)$: Construct the superposition: $|\pi\rangle = \sum_{x \in X} \sqrt{\pi_x}|x\rangle$;
- $\texttt{Update}(P)$: Apply any of $V(P)$, $V(P)^\dagger$ or $\textsc{Shift}$, where $V(P)$ satisfies $V(P)|x\rangle|\overline{0}\rangle = |x\rangle|p_x\rangle = |x\rangle \sum_{y \in X} \sqrt{p_{xy}}|y\rangle$, for all $x \in X$.

Implicitly, we also allow any controlled version of $\texttt{Check}(M)$, $\texttt{Setup}(P)$ and $\texttt{Update}(P)$, on which we access via oracle.

In terms of applications of $\textsc{Shift}$, `Update` has complexity 1, and, `Setup` has complexity $\mathrm{O}(\delta_G)$ (the diameter of G). Nonetheless, in many algorithmic applications, the situation is more complex and the number of applications of $\textsc{Shift}$ is not the only relevant cost, see for instance [1,2].

2.3 Classical Hitting Time

From now on we will assume that all the eigenvalues of P are within $[0, 1]$. This is without loss of generality by replacing P by $(\mathrm{Id} + P)/2$ if necessary. From the ergodicity of P, the eigenvalue 1 has multiplicity 1. The classical hitting time, $\mathrm{HT}(P, M)$, is defined as the expected number of applications of the Markov chain P required to hit a marked vertex when starting from π. This can be used to design a randomized algorithm for DETECT and FIND based on the corresponding random walk.

Proposition 1. *Let $k \geq 1$. $\textsc{Detect}^{(\geq k)}(G)$ can be solved with high probability and randomized complexity of order*

$$\mathsf{S} + T \times (\mathsf{U} + \mathsf{C}), \quad \textit{where} \quad T = \max_{|M'|=k} \mathrm{HT}(P, M').$$

$\textsc{Find}(G)$ can be solved with high probability and expected randomized complexity of order

$$\mathsf{S} + T \times (\mathsf{U} + \mathsf{C}), \quad \textit{where} \quad T = \mathrm{HT}(P, M).$$

Let P' be the Markov chain obtained from P by turning all outgoing transitions from marked vertices into self-loops. We call P' the *absorbing* version of P. If we arrange the elements of X so that the marked vertices are the last ones, matrices P and P' have the following block structure:

$$P := \begin{pmatrix} P_{UU} & P_{UM} \\ P_{MU} & P_{MM} \end{pmatrix}, \qquad P' := \begin{pmatrix} P_{UU} & P_{UM} \\ 0 & I \end{pmatrix}.$$

where P_{UU} and P_{MM} are matrices of size $(n-m) \times (n-m)$ and $m \times m$, while P_{UM} and P_{MU} are matrices of size $(n-m) \times m$ and $m \times (n-m)$.

We first present the matrix characterization of $\mathrm{HT}(P, M)$. From the reversibility of P, we get that $D(P) = \mathrm{diag}(\sqrt{\pi}) P \,\mathrm{diag}(\sqrt{\pi})^{-1}$ is a symmetric matrix, which is also known as the discriminant of P. The latter was introduced in [10] for symmetric P, and generalized to reversible P in [12]. Set $|\pi\rangle = \sum_{x \in X} \sqrt{\pi_x} |x\rangle$, $p_M = \sum_{x \in M} \pi_x$ and $p_U = \sum_{x \in X \setminus M} \pi_x$. The respective normalized projections of $|\pi\rangle$ on marked and unmarked vertices are $|M\rangle = \sum_{x \in M} \sqrt{\pi_x / p_M} |x\rangle$ and $|U\rangle = \sum_{x \in X \setminus M} \sqrt{\pi_x / p_U} |x\rangle$. Then

$$\mathrm{HT}(P, M) = p_U \times \langle U | (\mathrm{Id}_U - D(P)_{UU})^{-1} | U \rangle.$$

For simplicity, we will from now on omit the quantity p_U in the above definition. Indeed, we assume that $p_M \leq 1/2$, so that the the difference between the two expressions is at most a factor of 2. Note that if $p_M > 1/2$, then there is no need for any (classical or quantum) walk to find a marked vertex.

We now introduce the spectral characterization of $\mathrm{HT}(P, M)$. Note that the reversibility of P also implies the alternative definition for the discriminant

$$(D(P))_{xy} = \sqrt{p_{xy} p_{yx}}.$$

Extending the latter definition of the discriminant to P', we get that $D(P') = D(P)_{UU} \oplus \mathrm{Id}_M$. Let $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ be a system of orthonormal eigenvectors

of $D(P')$ with respective eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{n-m} < \lambda_{n-m+1} = \ldots = \lambda_n = 1$, so that $D(P') = \sum_k \lambda_k |v_k\rangle\langle v_k|$. Then one can rewrite $\mathrm{HT}(P, M)$ as:

$$\mathrm{HT}(P, M) = \sum_{k \leq n-m} \frac{|\langle v_k|U\rangle|^2}{1 - \lambda_k}.$$

2.4 Quantum Hitting Time

Quantum walks were successfully used for detecting the presence of marked vertices quadratically faster than P. Nonetheless, only little is known on the problem of finding a marked vertex. Below we illustrate the state of the art.

Theorem 1 ([10]). *Let $k \geq 1$. $\textsc{Detect}^{(\geq k)}(G)$ can be solved with high probability and quantum complexity of order*

$$\mathsf{S} + T \times (\mathsf{U} + \mathsf{C}), \quad \textit{where} \quad T = \max_{|M'|=k} \sqrt{\mathrm{HT}(P, M')}.$$

When P is state-transitive and there is a unique marked vertex z (i.e., $m = 1$), $\mathrm{HT}(P, \{z\})$ is independent of z and one can also find z:

Theorem 2 ([14,13]). *Assume that P is state-transitive. $\textsc{Find}^{(1)}(G)$ can be solved with high probability and quantum complexity of order*

$$\mathsf{S} + T \times (\mathsf{U} + \mathsf{C}), \quad \textit{where} \quad T = \sqrt{\mathrm{HT}(P, \{z\})}.$$

Using standard techniques, such as in [5], Theorem 2 can be generalized to any number of marked vertices, with an extra logarithmic multiplication factor. Nonetheless, the complexity of the corresponding algorithms does not decrease when the size of M increases, contrary to the random walk search algorithm (Proposition 1) and the quantum walk detecting algorithm (Theorem 1).

Corollary 1. *Assume that P is state-transitive. $\textsc{Find}(G)$ can be solved with high probability and quantum complexity of order*

$$\log(n) \times \Big(\mathsf{S} + T \times (\mathsf{U} + \mathsf{C})\Big), \quad \textit{where} \quad T = \sqrt{\mathrm{HT}(P, \{z\})}, \textit{ for any } z.$$

2.5 Szegedy's Quantum Analogue

Following the main lines of Szegedy [10], we define a quantum analogue of a reversible Markov chain P. Recall that $|\bar{0}\rangle$ is an arbitrary reference state in $\mathcal{H}$, and $V(P)|x\rangle|\bar{0}\rangle = |x\rangle|p_x\rangle$. Set $\mathcal{X} = \mathcal{H} \otimes \mathrm{span}\,\{|\bar{0}\rangle\} = \mathrm{span}\,\{|x\rangle|\bar{0}\rangle : x \in X\}$, and $\mathrm{ref}_{\mathcal{X}} = 2\sum_{x\in X} |x\rangle\langle x| \otimes |\bar{0}\rangle\langle\bar{0}| - \mathrm{Id}$, the reflection with respect to $\mathcal{X}$.

Definition 4. *The* quantum analogue *of P is*

$$W(P) = V(P)^\dagger \cdot \textsc{Shift} \cdot V(P) \cdot \mathrm{ref}_{\mathcal{X}},$$

and its walk space is the subspace spanned by $\mathcal{X}$ and $W(P)\mathcal{X}$.

Observe that in [10], the quantum walk is actually defined as $(V(P) \cdot W(P) \cdot V(P)^\dagger)^2$. Moreover, $W(P)$ requires 3 calls to $\texttt{Update}(P)$, and $\textsc{Shift}$ always succeeds when $W(P)$ is restricted to its walk space.

Szegedy proved the following useful lemma which relates the spectral decomposition of $W(P)$ in its walk space to the one of P. Let $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ be the normalized eigenvectors of $D(P)$ with respective eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{n-1} < \lambda_n = 1$. From the definition of $D(P)$, observe that $|\pi\rangle$ is a 1-eigenvector of $D(P)$, and therefore we set $|v_n\rangle = |\pi\rangle$.

Lemma 1 ([10]). *Define* $\mathcal{B}_k = \text{span}\{|v_k\rangle|\bar{0}\rangle, W(P)|v_k\rangle|\bar{0}\rangle\}$, *for* $k \neq n$, *and* $\mathcal{B}_n = \text{span}\{|v_n\rangle|\bar{0}\rangle\}$. *Then the walk space of* $W(P)$ *is* $\bigoplus_k \mathcal{B}_k$, *where* $W(P)$ *admits the following spectral decomposition:*

- *On* $\mathcal{B}_k$, $k \neq n$: $\mu_k^{\pm} = e^{\pm i\varphi_k}, |\Psi_k^{\pm}\rangle$, *where* $\cos\varphi_k = \lambda_k$ *and* $\frac{|\Psi_k^+\rangle + |\Psi_k^-\rangle}{\sqrt{2}} = |v_k\rangle|\bar{0}\rangle$
- *On* $\mathcal{B}_n$: $\mu_n = 1, |\Psi_n\rangle = |v_n\rangle|\bar{0}\rangle$. $\mathcal{B}^{\perp}$).

Therefore, we will also call $|\Psi_n\rangle = |v_n\rangle|\bar{0}\rangle$ the *stationary distribution* of $W(P)$.

3 Finding via Quantum Walk

3.1 Classical Interpolation

Our starting state is the stationary superposition $|v_n\rangle|\bar{0}\rangle = |\pi\rangle|\bar{0}\rangle$ of $W(P)$. We would like to end in its projection onto marked vertices, namely $|M\rangle|\bar{0}\rangle$, which is also the stationary superposition of $W(P')$. However, in many cases, including the 2D-grid, every iteration of $W(P')$ on $|\pi\rangle|\bar{0}\rangle$ may remain far from $|M\rangle|\bar{0}\rangle$.

Our approach consists in taking a new random walk, namely an interpolation between P and P', This technique is drastically different from the approach of [14,13], and up to our knowledge new.

Definition 5. *The* classical interpolation *of* P *and* P' *is*

$$P(s) = (1-s)P + sP', \quad 0 \leq s \leq 1.$$

This interpolation has some similarities with adiabatic evolutions, where similar interpolations are usually defined for Hamiltonians. Here the interpolation is of different nature after we take the quantum analogue of $P(s)$. Nonetheless, this interpolation still makes sense for adiabatic evolutions, leading to an interesting connection between the hitting time and the adiabatic condition [15].

Note that $P(0) = P$, $P(1) = P'$, and $P(s)$ has block structure

$$P(s) = \begin{pmatrix} P_{UU} & P_{UM} \\ (1-s)P_{MU} & (1-s)P_{MM} + sI \end{pmatrix}.$$

Decompose the stationary distribution of P as $\pi = \begin{pmatrix} \pi_U & \pi_M \end{pmatrix}$. Remember that $p_M = \sum_{x \in M} \pi_x$ is the probability to pick a marked vertex from the stationary distribution. Then

$$\pi(s) = \frac{1}{1-(1-s)p_M} \begin{pmatrix} (1-s)\pi_U & \pi_M \end{pmatrix}$$

is a stationary distribution of $P(s)$, for $s \in [0,1]$. Moreover one can show that:

Fact 1. *For* $s \in [0,1)$, *the Markov chain* $P(s)$ *is ergodic and reversible.*

For $s \in [0,1)$, let $D(s)$ be the discriminant of $P(s)$. Then $D(s)$ and $P(s)$ are similar and therefore have the same eigenvalues. Let $|v_1(s)\rangle, |v_2(s)\rangle, \ldots, |v_n(s)\rangle$

be a system of orthonormal eigenvectors of $D(s)$ with respective eigenvalues $0 \leq \lambda_1(s) \leq \lambda_2(s) \leq \ldots \leq \lambda_{n-1}(s) < \lambda_n(s) = 1$: $D(s) = \sum_k \lambda_k(s)|v_k(s)\rangle\langle v_k(s)|$. We also define $W(s) = W(P(s))$.

For $s = 1$, we extend the above for $P(1) = P'$. Recall that $|v_n(s)\rangle = |\pi(s)\rangle$ is an eigenvector of $D(s)$ with eigenvalue $\lambda_n(s) = 1$. Observe that $|v_n(s)\rangle$ is in the two-dimensional subspace spanned by $|M\rangle$ and $|U\rangle$.

Fact 2. $|v_n(s)\rangle = \cos\theta(s)|U\rangle + \sin\theta(s)|M\rangle$, *where* $\theta(s) = \arcsin\sqrt{\frac{p_M}{1-s(1-p_M)}}$.

Intuitively we want $|v_n(s)\rangle$ to have a large overlap on both $|U\rangle$ and $|M\rangle$, so that the algorithm will proceed in two steps: first, map $|U\rangle$ to $|v_n(s)\rangle$, using the quantum walk (using `Update`); second, map $|v_n(s)\rangle$ to $|M\rangle$ by projecting onto marked vertices (using `Check`). Therefore, ideally we want s to satisfy $\sin\theta(s) = \cos\theta(s) = 1/\sqrt{2}$, namely $s = s(p_M)$, where $s(p_M) = 1 - \frac{p_M}{1-p_M}$.

3.2 Quantum Circuit for $W(s)$

In the following lemma, we assume to know p_{xx} for every x. This is reasonable since in practice the probability of self-loops is known. In many cases, it is even independent of x. For the rest of the paper, we assume that this is not an obstacle (we can assume that one call to $\texttt{Update}(P)$ allows to learn p_{xx} for any x).

Lemma 2. *Assuming that p_{xx} is known for every x,* $\texttt{Update}(P(s))$ *can be implemented with complexity* $\mathsf{C} + \mathsf{U}$.

Proof. From Definition 4, the quantum analogue of $P(s)$ is $W(s) = V(P(s))^\dagger \cdot \textsc{Shift} \cdot V(P(s)) \cdot \mathrm{ref}_{\mathcal{X}}$, where $V(P(s))$ is a unitary that maps $|x\rangle|\bar{0}\rangle$ to $|x\rangle|p_x(s)\rangle = |x\rangle\sum_{y\in X}\sqrt{p_{xy}(s)}|y\rangle$. Since Shift only depends on G, we just need to explain how to implement $V(P(s))$ and its inverse. We now explain how to implement $V(P(s))$ using one call to $V(P)$ and 2 calls to $\texttt{Check}(M)$. The algorithm for its inverse is obtained from the reverse algorithm.

Simulation(P, M, s)

1. Let $|x\rangle|\bar{0}\rangle$ be the current state
2. Use a fresh qubit (*marking* register) in state $|0\rangle$ and call $\texttt{Check}(M)$.
3. If the marking register is in state $|0\rangle$, call $V(P)$: $|x\rangle|p_{xy}\rangle|0\rangle$
4. Otherwise
   (a) Use another fresh qubit in state $|0\rangle$: $|x\rangle|\bar{0}\rangle|1\rangle|0\rangle$
   (b) Apply a rotation of angle $\arcsin\sqrt{s}$ on the fourth register: $|x\rangle|\bar{0}\rangle|1\rangle(\sqrt{1-s}|0\rangle + \sqrt{s}|1\rangle)$
   (c) If the fourth register is $|0\rangle$, apply $V(P)$ on the first two registers, Otherwise XOR the first two registers: $|x\rangle(\sqrt{1-s}|p_x\rangle|1\rangle|0\rangle + \sqrt{s}|x\rangle|1\rangle|1\rangle)$
   (d) If the second register is $|x\rangle$, apply a rotation of angle $-\arcsin\sqrt{s/((1-s)p_{xx}+s)}$ on the fourth register, Otherwise do nothing: $|x\rangle|p_x(s)\rangle|1\rangle|0\rangle$
5. Call $\texttt{Check}(M)$ to uncompute the marking register.

□

3.3 Hitting Time in s

Following [15], we define the hitting time in s, which intuitively corresponds to the expected time for $P(s)$ to converge to $\pi(s)$ from $\pi(0)$.

Definition 6. $\mathrm{HT}(s) = \sum_{k \neq n} \frac{|\langle v_k(s)|U\rangle|^2}{1-\lambda_k(s)}$.

In particular, note that $\mathrm{HT}(1) = \mathrm{HT}(P, M)$ is the usual hitting time of P with respect to the set of marked vertices M.

The running time of our quantum search algorithms will depend on $\mathrm{HT}(s)$ for some particular value $s \in [0, 1]$. This can be related to the usual hitting time $\mathrm{HT}(P, M)$ thanks to the following explicit expression for $\mathrm{HT}(s)$ (see [15]).

Theorem 3. $\mathrm{HT}(s) = \sin^4 \theta(s) \cdot \mathrm{HT}(P, M)$.

4 Quantum Search

4.1 Algorithm with Known Parameters

Theorem 4 (Phase estimation [16,17]). *Let W be a unitary operator on $\mathcal{H}$ and $t \in \mathbb{N}$. There exists a quantum circuit* **PhaseEstimation**(W, t) *using 2^t calls to the controlled operator $\mathrm{c}{-}W$ and $\mathrm{O}(t^2)$ additional gates, and acting on eigenstates $|\Psi_k\rangle$ of W as*

$$|\Psi_k\rangle \to |\Psi_k\rangle \frac{1}{2^t} \sum_{l,m=0}^{2^t-1} e^{-\frac{2\pi i l m}{2^t}} e^{i\varphi_k l} |m\rangle,$$

where $e^{i\varphi_k}$ is the eigenvalue of W corresponding to $|\Psi_k\rangle$.

By linearity, Theorem 4 implies that **PhaseEstimation**(W, t) resolves any state along the eigenstates of W, labelling those states with a second register whose measurement yields an approximation of the first t-bit of the binary decomposition of $\varphi_k/(2\pi)$. Here, we will be mostly interested in the $|\Psi_n\rangle$-component, with corresponding phase $\varphi_n = 0$. In that case, the second register is in the state $|0^t\rangle$ and the estimation is exact.

We define our main search algorithm with parameters $0 \leq p^* \leq 1$ (an approximation of p_M) and $t \in \mathbb{N}$. Recall that $s(p^*) = 1 - \frac{p^*}{1-p^*}$. For suitable parameters, the algorithm outputs a marked vertex with high probability, if there is any.

QuantumWalkSearch(P, M, p^*, t)
1. Prepare the state $|\pi\rangle|\bar{0}\rangle$
2. Use a fresh qubit in state $|0\rangle$, apply `Check`(M) and measure the qubit.
3. If the outcome is $|1\rangle$, measure the first register (in the vertex basis) and output the outcome.
4. Otherwise, apply **PhaseEstimation**$(W(s(p^*)), t)$ on the registers.
5. Use a fresh qubit in state $|0\rangle$, apply `Check`(M) and measure the qubit.
6. If the outcome is $|1\rangle$, measure the first register (in the vertex basis) and output the outcome.
7. Otherwise, output "`No marked vertex`"

Theorem 5. *Let* $p^*, \epsilon_1 \in [0,1]$ *be such that* $\cos^2\theta(s)\sin^2\theta(s) \geq \epsilon_1$, *where* $s = s(p^*)$. *Let* $T \geq 1$ *and* $\epsilon_2 \in [0,1]$ *be such that* $T \geq \frac{\pi}{\sqrt{2\epsilon_2}} \times \sqrt{\mathrm{HT}(s)}$. *Then,* **QuantumWalkSearch**$(P, M, p^*, \lceil \log T \rceil)$ *outputs a marked vertex with probability at least* $\epsilon_1 - \epsilon_2$ *and complexity of order* $\mathsf{S} + T \times (\mathsf{U} + \mathsf{C})$.
In particular, if $|p^* - p_M| \leq p_M/3$ *and* $T \geq 10\sqrt{\mathrm{HT}(P,M)}$, *then the success probability is at least* $1/20$.

Proof. Let $t = \lceil \log T \rceil$. First observe that the complexity analysis is direct since **QuantumWalkSearch**(P, M, p^*, t) has complexity of order $\mathsf{S} + 2^t \times (\mathsf{U} + \mathsf{C})$. We now assume that we reach Step 4. Otherwise a marked vertex is already found. Then the current state before Step 4 is $|U\rangle|\bar{0}\rangle$. Let $\alpha_k(s) = \langle U|v_k(s)\rangle$. From now on, we omit to write the dependence on s explicitly, when there is no ambiguity.

In Step 4, **PhaseEstimation**$(W(s), t)$ is applied on the state

$$|U\rangle|\bar{0}\rangle = \alpha_n|v_n\rangle|\bar{0}\rangle + \sum_{k\neq n}\alpha_k|v_k\rangle|\bar{0}\rangle = \alpha_n|\Psi_n\rangle + \frac{1}{\sqrt{2}}\sum_{k\neq n}\alpha_k\left(|\Psi_k^+\rangle + |\Psi_k^-\rangle\right).$$

Theorem 4 shows that **PhaseEstimation**$(W(s), t)$ maps $|\Psi_n\rangle$ to $|\Psi_n\rangle|0^t\rangle$ and maps $|\Psi_k^\pm\rangle$ to $|\Psi_k^\pm\rangle\left(\delta_k^\pm|0^t\rangle + |\eta_k^\pm\rangle\right)$, where

$$\delta_k^\pm = \frac{1}{2^t}\sum_{l=0}^{2^t-1} e^{\pm i\varphi_k l} \quad \text{and} \quad |\eta_k^\pm\rangle = \frac{1}{2^t}\sum_{m=1}^{2^t-1}\sum_{l=0}^{2^t-1} e^{-\frac{2\pi i l m}{2^t}} e^{\pm i\varphi_k l}|m\rangle.$$

By definition, $\langle 0^t|\eta_k^\pm\rangle = 0$. Then, the probability p to obtain a marked vertex by measuring the first register is at least the probability to obtain both a marked vertex in the first register and the state $|0^t\rangle$ in the last register (i.e., the phase is estimated to be 0). Since $|\Psi_n\rangle = |v_n\rangle|\bar{0}\rangle$ and using Fact 2, we see that the probability p is lower bounded as

$$p \geq |\alpha_n|^2\|\Pi_M|v_n\rangle\|^2 - \frac{1}{2}\sum_{k\neq n}|\alpha_k|^2\left(|\delta_k^+|^2 + |\delta_k^-|^2\right) = \cos^2\theta\sin^2\theta - \sum_{k\neq n}|\alpha_k|^2\delta_k^2,$$

where $\Pi_M = \sum_{x\in M}|x\rangle\langle x|$ is the projection onto marked vertices and $\delta_k = |\delta_k^+| = |\delta_k^-|$. By hypothesis, we already have $\cos^2\theta\sin^2\theta \geq \epsilon_1$. Therefore, it remains to prove that the second term in the RHS is at least $-\epsilon_2$.

First, using the definition of δ_k, we get: $\delta_k^2 = \frac{\sin^2(2^{t-1}\varphi_k)}{2^{2t}\sin^2(\varphi_k/2)} \leq \frac{\pi^2}{2^{2t}\varphi_k^2}$. We also have by definition of $\mathrm{HT}(s)$:

$$\mathrm{HT}(s) = \sum_{k\neq n}\frac{|\alpha_k|^2}{1-\cos\varphi_k} = \sum_{k\neq n}\frac{|\alpha_k|^2}{2\sin^2(\varphi_k/2)} \geq 2\sum_{k\neq n}\frac{|\alpha_k|^2}{\varphi_k^2},$$

which together with the above implies that $\sum_{k\neq n}|\alpha_k|^2\delta_k^2 \leq \frac{\pi^2}{2}\frac{\mathrm{HT}(s)}{2^{2t}} \leq \epsilon_2$.

We now prove the last part of the theorem. The following fact is easy to prove:

Fact 3. *Let* $\epsilon_1 \leq 1/4$ *be such that* $2\sqrt{\epsilon_1}p_M \leq p^* \leq 2(1-\sqrt{\epsilon_1})p_M$. *Then,* $\cos^2\theta(s)\sin^2\theta(s) \geq \epsilon_1$.

The conditions of Fact 3 are satisfied with $\epsilon_1 = 1/10$. Set $\epsilon_2 = 1/20$. Using $\mathrm{HT}(s) \leq \mathrm{HT}(P,M)$, one can check that the conditions of the theorem are satisfied, and therefore the success probability is at least $\epsilon_1 - \epsilon_2 = 1/20$. □

4.2 General Case

At first, assume we have a correct approximation p^* of p_M. In that case, even if we do not know $\mathrm{HT}(P, M)$, we can use the following algorithm, and still find a marked vertex with an expected cost $\mathrm{O}(\sqrt{\mathrm{HT}(P, M)})$.

QuantumWalkSearch$'(P, M, p^*, k)$
1. Let $t = 1$.
2. Call k times **QuantumWalkSearch**(P, M, p^*, t).
3. If no marked vertex is found, set $t \leftarrow t + 1$ and go back to step 2.

Theorem 6. *Given* p^* *such that* $|p^* - p_M| \leq p_M/3$, **QuantumWalkSearch**$'(P, M, p^*, 28)$ *solves* $\textsc{Find}(G)$ *with expected quantum complexity of order*

$$\log(T) \times \mathsf{S} + T \times (\mathsf{U} + \mathsf{C}), \quad \textit{where} \quad T = \sqrt{\mathrm{HT}(P, M)}.$$

Proof. The general idea is to use **QuantumWalkSearch**(P, M, p^*, t) with increasing accuracy of the phase estimation (parameter t), until it is high enough so that the algorithm outputs a marked element with high probability.

Set $s = s(p^*)$. Let t_0 to be the integer such that $\pi\sqrt{\frac{\mathrm{HT}(s)}{2\epsilon_2}} \leq 2^{t_0} \leq \pi\sqrt{\frac{2\,\mathrm{HT}(s)}{\epsilon_2}}$, and let $T = 2^{t_0} = \mathrm{O}(\sqrt{\mathrm{HT}(s)})$. By Theorem 5, for any $t \geq t_0$, **QuantumWalkSearch**(P, M, p^*, t) outputs a marked vertex with probability at least $1/20$. Then, step 3 is reached without finding any marked vertex with probability at most $p \leq (1 - 1/20)^{28} \leq 1/4$. Moreover, **QuantumWalkSearch**(P, M, p^*, t) has complexity of order $\mathsf{S} + 2^t \times (\mathsf{U} + \mathsf{C})$.

Let t_f be the value of t when **QuantumWalkSearch**$'(P, M, p^*, 28)$ stops, that is, the number of iterations of step 2. Then, the expected complexity of **QuantumWalkSearch**$'(P, M, p^*, 28)$ is of order $N_1 \times \mathsf{S} + N_2 \times (\mathsf{U} + \mathsf{C})$, where N_1 is the expectation of t_f, and N_2 is the expectation of $2 + 4 + \ldots + 2^{t_f}$.

First observe that $N_1 \leq t_0 + \sum_{t=t_0+1}^{\infty} p^{t-t_0} = \mathrm{O}(t_0)$. For N_2 we get

$$N_2 \leq \sum_{t=1}^{t_0} 2^t + \sum_{t=t_0+1}^{\infty} p^{t-t_0} \cdot 2^t = (2 \cdot 2^{t_0} - 2) + 2^{t_0} \sum_{t=1}^{\infty} p^t \cdot 2^t.$$

Then using the fact that $p \leq 1/4$ we finally obtain

$$N_2 \leq 2 \cdot 2^{t_0} + 2^{t_0} \sum_{t=1}^{\infty} 2^{-t} \leq 3 \cdot 2^{t_0}.$$

This concludes the proof since $2^{t_0} = \mathrm{O}(\sqrt{\mathrm{HT}(s)})$ and $\mathrm{HT}(s) \leq \mathrm{HT}(P, M)$. □

For the general case we get two possible situations, depending on whether a lower bound $p_{\min}$ on p_M and/or an upper bound $\mathrm{HT}_{\max}$ on $\mathrm{HT}(P, M)$ is given. In particular, for $\textsc{Find}(G)^{(\geq k)}$, we can set $p_{\min} = \min_{M':|M'|=k} p_{M'}$ and $\mathrm{HT}_{\max} = \max_{M':|M'|=k} \mathrm{HT}(P, M')$.

Theorem 7. *Given* $p_{\min} \leq p_M$, $\textsc{Find}(G)$ *can be solved with expected quantum complexity of order*

$$\sqrt{\log(1/p_{\min})} \times \big[\log(T) \times \mathsf{S} + T \times (\mathsf{U} + \mathsf{C})\big], \quad \textit{where} \quad T = \sqrt{\mathrm{HT}(P, M)}.$$

Moreover, if $\mathrm{HT}_{\max} \geq \mathrm{HT}(P, M)$ *is also given, then* $\textsc{Find}(G)$ *can be solved with expected quantum complexity of order*

$$\sqrt{\log(1/p_{\min})} \times \big[\mathsf{S} + T \times (\mathsf{U} + \mathsf{C})\big], \quad \textit{where} \quad T = \sqrt{\mathrm{HT}_{\max}}.$$

Proof. We simply prove the first part of the theorem. The second one is similar using $\mathbf{QuantumWalkSearch}(P, M, p^*, T)$ instead of $\mathbf{QuantumWalkSearch}'(P, M, p^*, 28)$.

From Theorem 6, it is enough to have a good approximation p^* of p_M, such that $3p^*/4 \leq p_M \leq 3p^*/2$. Moreover, since $p_{\min} \leq p_M \leq 1/2$, this condition will be satisfied for some $p^* \in \{(2/3) \times 2^{-l} : l = 1, \ldots, \lfloor \log(1/p_{\min}) \rfloor\}$.

Let us incorporate step 2 of $\mathbf{QuantumWalkSearch}'(P, M, p^*, 28)$ into a loop on the $\lfloor \log(1/p_{\min}) \rfloor$ possible values of p^*. Then the analysis is basically the same, except that now the complexity of step 2 is multiplied by a factor of order $\log(1/p_{\min})$. Instead of looping on all possible values of p^*, we can search for the right value using Grover's algorithm, following the approach of [18], therefore reducing the multiplication factor to $\sqrt{\log(1/p_{\min})}$. □

Theorem 8. *Given* $\mathrm{HT}_{\max} \geq \mathrm{HT}(P, M)$, $\textsc{Find}(G)$ *can be solved with expected quantum complexity of order*

$$\log(1/p_M) \times \big[\mathsf{S} + T \times (\mathsf{U} + \mathsf{C})\big], \quad \textit{where} \quad T = \sqrt{\mathrm{HT}_{\max}}.$$

Proof. We now use $\mathbf{QuantumWalkSearch}(P, M, p^*, t)$ with $t = \lceil \log \sqrt{\mathrm{HT}_{\max}} \rceil$, and perform a dichotomic search for an appropriate value of p^*. This dichotomic search uses backtracking since the branching in the dichotomy is with bounded error, similarly to the situation in [19].

Initially we set $a = 0$ and $b = 1$. Then for testing the current value of $p^* = (a + b)/2$, we run a constant number of times $\mathbf{QuantumWalkSearch}(P, M, p^*, t)$. If a marked vertex is found we stop. Otherwise, if $\mathbf{PhaseEstimation}(W(s(p^*)), t)$ outputs a minority of 0s, we set $a = p^*$, otherwise we set $b = p^*$. The details of the analysis are given in [19]. □

4.3 Application to the 2D-Grid

Consider a random walk on the 2D-grid of size $\sqrt{n} \times \sqrt{n}$, with self-loops. In this section we consider only the complexity in terms of the number of uses of `Check` and $\textsc{Shift}$. The previous best known quantum complexity of $\textsc{Find}(G)^{(k)}$ and $\textsc{Find}(G)^{(\geq k)}$ was $\mathrm{O}(\sqrt{n}(\log n)^{3/2})$, from Corollary 1. Since the grid is a 5-regular graphs (4 directions and 1 self-loop), P is symmetric, and therefore the stationary distribution of P is uniform, and we simply have $p_M = m/n$. Then `Setup` is realized with $\sqrt{n}$ uses of $\textsc{Shift}$, and $\mathrm{HT}(P, \{z\}) = \Theta(n \log n)$, for any z. Therefore we get the following corollary of Theorem 5 and Theorem 7, by upper bounding $\mathrm{HT}(P, M) = \mathrm{O}(n \log n)$.

Corollary 2. *Let* G *be the* 2D*-grid of size* $\sqrt{n} \times \sqrt{n}$*, and let* $k \geq 1$*. Then* $\textsc{Find}(G)^{(k)}$ *can be solved with expected quantum complexity* $\mathrm{O}(\sqrt{n \log n})$*, and* $\textsc{Find}(G)^{(\geq k)}$ *with expected quantum complexity* $\mathrm{O}(\sqrt{n \times \log n \times \log(n/k)})$.

Acknowledgments

This research has been supported in part by ARO/NSA under grant W911NF-09-1-0569. M. Ozols acknowledges support from QuantumWorks. F. Magniez acknowledges support from French ANR grants CRYQ (ANR-09-JCJC-0067) and QRAC (ANR-08-EMER-012), as well as the European Commission IST Integrated Project Qubit Applications (QAP) 015848.

References

1. Ambainis, A.: Quantum walk algorithm for Element Distinctness. In: Proc. 45th FOCS, pp. 22–31. IEEE Computer Society Press, New York (2004)
2. Magniez, F., Santha, M., Szegedy, M.: Quantum Algorithms for the Triangle Problem. In: Proc. 16th SODA (2005)
3. Buhrman, H., Špalek, R.: Quantum verification of matrix products. In: Proc. 17th ACM-SIAM Symposium on Discrete Algorithms, pp. 880–889 (2006)
4. Magniez, F., Nayak, A.: Quantum complexity of testing group commutativity. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1312–1324. Springer, Heidelberg (2005)
5. Aaronson, S., Ambainis, A.: Quantum search of spatial regions. Theory of Computing 1, 47–79 (2005)
6. Shenvi, N., Kempe, J., Whaley, K.: Quantum random-walk search algorithm. Phys. Rev. A 67, Article no. 052307 (2003)
7. Childs, A.M., Goldstone, J.: Spatial search and the Dirac equation. Phys. Rev. A 70(4), 042312 (2004)
8. Ambainis, A., Kempe, J., Rivosh, A.: Coins make quantum walks faster. In: Proc. 16th SODA, pp. 1099–1108 (2005)
9. Kempe, J.: Discrete quantum walks hit exponentially faster. Prob. Th. Rel. Fields 133(2), 215–235 (2005)
10. Szegedy, M.: Quantum speed-up of Markov chain based algorithms. In: Proc. 45th FOCS, pp. 32–41 (2004)
11. Krovi, H., Brun, T.A.: Hitting time for quantum walks on the hypercube. Phys. Rev. A 73(3), 032341 (2006)
12. Magniez, F., Nayak, A., Roland, J., Santha, M.: Search via quantum walk. In: Proc. 39th STOC, pp. 575–584. ACM Press, New York (2007)
13. Magniez, F., Nayak, A., Richter, P.C., Santha, M.: On the hitting times of quantum versus random walks. In: Proc. 19th SODA, SIAM, pp. 86–95. SIAM, Philadelphia (2009)
14. Tulsi, A.: Faster quantum-walk algorithm for the two-dimensional spatial search. Phys. Rev. A 78(1), 012310 (2008)
15. Krovi, H., Ozols, M., Roland, J.: On the adiabatic condition and the quantum hitting time of Markov chains. Technical Report arXiv:1004.2721, arXiv.org (2010)
16. Kitaev, A.: Quantum measurements and the Abelian stabilizer problem. Technical Report quant-ph/9511026, arXiv.org (1995)
17. Cleve, R., Ekert, A., Macchiavello, C., Mosca, M.: Quantum algorithms revisited. Proc. Royal Society A 454(1969), 339–354 (1998)
18. Høyer, P., Mosca, M., de Wolf, R.: Quantum search on bounded-error inputs. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 291–299. Springer, Heidelberg (2003)
19. Feige, U., Raghavan, P., Feleg, D., Upfal, E.: Computing with noisy information. SIAM Journal on Computing 23(5), 1001–1018 (1994)

Improved Constructions for Non-adaptive Threshold Group Testing

Mahdi Cheraghchi*

School of Computer and Communication Sciences
EPFL, 1015 Lausanne, Switzerland
mahdi.cheraghchi@epfl.ch

Abstract. The basic goal in combinatorial group testing is to identify a set of up to d defective items within a large population of size $n \gg d$ using a pooling strategy. Namely, the items can be grouped together in pools, and a single measurement would reveal whether there are one or more defectives in the pool. The threshold model is a generalization of this idea where a measurement returns positive if the number of defectives in the pool passes a fixed threshold u, negative if this number is below a fixed lower threshold $\ell \leq u$, and may behave arbitrarily otherwise. We study non-adaptive threshold group testing (in a possibly noisy setting) and show that, for this problem, $O(d^{g+2}(\log d)\log(n/d))$ measurements (where $g := u - \ell$) suffice to identify the defectives, and also present almost matching lower bounds. This significantly improves the previously known (non-constructive) upper bound $O(d^{u+1}\log(n/d))$. Moreover, we obtain a framework for explicit construction of measurement schemes using lossless condensers. The number of measurements resulting from this scheme is ideally bounded by $O(d^{g+3}(\log d)\log n)$. Using state-of-the-art constructions of lossless condensers, however, we come up with explicit testing schemes with $O(d^{g+3}(\log d)\mathsf{quasipoly}(\log n))$ and $O(d^{g+3+\beta}\mathsf{poly}(\log n))$ measurements, for arbitrary constant $\beta > 0$.

1 Introduction

Combinatorial group testing is a classical problem that deals with identification of sparse Boolean vectors using disjunctive queries. Suppose that among a large set of n items it is suspected that, for some *sparsity parameter* $d \ll n$, up to d items might be "defective". In technical terms, defective items are known as *positives* and the rest are called *negatives*. In a *pooling strategy*, the items can be arbitrarily grouped in pools, and a single "measurement" reveals whether there is one or more positives within the chosen pool. The basic goal in group testing is to design the pools in such a way that the set of positives can be identified within a number of measurements that is substantially less than n. Since its introduction in 1940's [1], group testing and its variations have been extensively studied and have found surprisingly many applications in seemingly unrelated

* Research supported by the ERC Advanced Investigator Grant 228021 of A. Shokrollahi.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 552–564, 2010.

areas. We refer the reader to [2,3] for an extensive review of major results in this area.

Formally, in classical group testing one wishes to learn an unknown d-sparse[1] Boolean vector $(x_1, \ldots, x_n) \in \{0,1\}^n$ using a set of m measurements, where each measurement is defined by a subset of the coordinates $\mathcal{I} \subseteq [n]$ and outputs the logical "or" $\bigvee_{i \in \mathcal{I}} x_i$. The basic goal is to design the measurements in such a way that all d sparse vectors become uniquely identifiable using as few measurements as possible. A natural generalization of classical group testing, introduced by Damaschke [4], considers the case where the measurement outcomes are determined by a *threshold predicate* instead of logical or. Namely, this model is characterized by two integer parameters ℓ, u such that $0 < \ell \leq u$ (that are considered to be fixed constants), and each measurement outputs positive if the number of positives within the corresponding pool is at least u. On the other hand, if the number of positives is less than ℓ, the test returns negative, and otherwise the outcome can be arbitrary. In this view, classical group testing corresponds to the special case where $\ell = u = 1$. In addition to being of theoretical interest, the threshold model is interesting for applications, in particular in biology, where the measurements have reduced or unpredictable sensitivity or may depend on various factors that must be simultaneously present in the sample.

The difference $g := u - \ell$ between the thresholds is known as the *gap* parameter. As shown by Damaschke [4], in threshold group testing identification of the set of positives is only possible when the number of positives is at least u. Moreover, regardless of the number of measurements, in general the set of positives can only be identified within up to g false positives and g false negatives (thus, unique identification can be guaranteed only when $\ell = u$). Additionally, Damaschke constructed a scheme for identification of the positives in the threshold model. For the gap-free case where $g = 0$, the number of measurements in this scheme is $O((d + u^2) \log n)$, which is nearly optimal (within constant factors). However, when $g > 0$, the number of measurements becomes $O(dn^b + d^u)$, for an arbitrary constant $b > 0$, if up to $g + (u-1)/b$ misclassifications are allowed. Moreover, Chang et al. [5] have proposed a different scheme for the gap-free case that achieves $O(d \log n)$ measurements. A drawback of both schemes mentioned here is that the measurements are adaptive, which makes them less viable for numerous applications, in particular, molecular biology. In a *non-adaptive* setting, all measurements must be specified before their outcomes are revealed. This makes it convenient to think of the measurements in a matrix form. Specifically, a non-adaptive *measurement matrix* is an $m \times n$ Boolean matrix whose ith row is the characteristic vector of the set of items participating in the ith pool, and the goal would be to design a suitable measurement matrix.

Recently, non-adaptive threshold testing has been considered by Chen and Fu [6]. They observe that, a generalization of the standard notion of disjunct matrices (the latter being extensively used in the literature of classical group testing) is suitable for the threshold model. In our work, we refer to this generalized notion as *strongly disjunct* matrices and to the standard notion as

[1] A vector is d-sparse if its support has size at most d.

classical disjunct matrices. Using strongly disjunct matrices, they show that $O(ed^{u+1}\log(n/d))$ non-adaptive measurements suffices to identify the set of positives (within g false positives/negatives) even if up to e erroneous measurements are allowed in the model. This number of measurements almost matches (up to constant factors) the known lower bounds on the number of rows of strongly disjunct matrices. However, the dependence on the sparsity parameter is d^{u+1}, which might be prohibitive for an interesting range of parameters, when the thresholds are not too small (e.g., $\ell = u = 10$) and the sparsity parameter is rather large (e.g., $d \approx n^{1/10}$).

In this work, we consider the non-adaptive threshold model in a possibly noisy setting, where a number of measurement outcomes (specified by an *error parameter* $e \geq 0$) may be incorrect. Our first observation is that, a new variation of classical disjunct matrices (that is strictly weaker than strongly disjunct matrices) suffices for the purpose of threshold group testing. Moreover, we show that this weaker notion is necessary as well, and thus, essentially captures the combinatorial structure of the threshold model. Using a probabilistic construction (that requires $\mathsf{poly}(d, \log n)$ bits of randomness), we construct generalized disjunct matrices with $O(d^{g+2}(\log d)\log(n/d))$ rows. Thus, we bring the exponent of d in the asymptotic number of measurements from $u+1$ (that is necessary for strongly disjunct matrices) down to $g+2$, which is *independent* of the actual choice of the thresholds and only depends on the *gap* between them. We also show that this tradeoff is essentially optimal.

We proceed to define a new auxiliary object, namely the notion of *regular* matrices, that might be of independent interest and turns out to be the key combinatorial object in our explicit constructions. Intuitively, given a gap $g \geq 0$, a suitable regular matrix M_1 can be used to "lift" any measurement matrix M_2 designed for the threshold model with lower threshold $\ell = 1$ and higher threshold $u = g+1$ up to any arbitrary lower threshold $\ell' > 1$ and the same gap g. Therefore, for instance, in order to address the gap-free model, it would suffice to have a non-adaptive scheme for the classical group testing model with $\ell = u = 1$. This transformation is accomplished using a simple product that increases the height of the original matrix M_2 by a multiplicative factor equal to the height of the regular matrix M_1, while preserving the distinguishing properties of the original matrix M_2. We will then introduce a framework for construction of regular matrices using *strong lossless condensers* that are fundamental objects in derandomization theory, and more generally, theoretical computer science. We show that, by using an optimal condenser, it is possible to construct regular matrices with only $O(d(\log d)\log n)$ rows. This almost matches the upper bound achieved by a probabilistic construction that we will also present. To this date, no explicit construction of such optimal lossless condensers is known (though probabilistic constructions are easy to come up with). However, using state of the art in explicit condensers [7,8], we will come up with two explicit constructions of regular matrices that achieve an almost linear dependence on d. By combining regular matrices with strongly disjunct ones we obtain our threshold testing schemes, achieving bounds that are summarized in Table 1. Due to space restrictions, in

Table 1. Summary of the parameters achieved by various threshold testing schemes. The noise parameter $p \in [0, 1)$ is arbitrary, and thresholds $\ell, u = \ell + g$ are fixed constants. "Exp" and "Rnd" respectively indicate explicit and randomized constructions. "KS" refers to the construction of strongly disjunct matrices based on Kautz-Singleton superimposed codes [9] when instantiated by different families of codes.

	Number of rows	Tolerable errors	Remarks
M1	$O(d^{u+1}\frac{\log(n/d)}{(1-p)^2})$	$\Omega(pd\frac{\log(n/d)}{(1-p)^2})$	Rnd: Random strongly disjunct matrices.
M2	$O((\frac{d}{1-p})^{u+1}\log n)$	$\Omega(pd\frac{\log n}{1-p})$	Exp: KS using codes on the Gilbert-Varshamov bound as constructed in [10].
M3	$O((\frac{d\log n}{1-p})^{u+1})$	$\Omega(pd\frac{\log n}{1-p})$	Exp: KS using Reed-Solomon codes.
M4	$O((\frac{d}{1-p})^{2u+1}\log n)$	$\Omega(pd\frac{\log n}{1-p})$	Exp: KS using Algebraic Geometric codes of Tsfasman-Vlăduţ-Zink [11]
M5	$O((\frac{d\sqrt{\log n}}{1-p})^{u+3/2})$	$\Omega(p(\frac{d\sqrt{\log n}}{1-p})^{3/2})$	Exp: KS using Hermitian codes ($d \gg \sqrt{\log n}$).
M6	$O(d^{g+2}\frac{(\log d)\log(n/d)}{(1-p)^2})$	$\Omega(pd\frac{\log(n/d)}{(1-p)^2})$	Rnd: Construction 3.
M7	$O(d^{g+3}\frac{(\log d)\log^2 n}{(1-p)^2})$	$\Omega(pd^2\frac{\log^2 n}{(1-p)^2})$	Constructions 5 and 2 combined, assuming optimal condensers and strongly disjunct matrices.
M8	$O(d^{g+3}\frac{(\log d)T_2\log n}{(1-p)^{g+2}})$	$\Omega(pd^2\frac{T_2\log n}{1-p})$	Exp: Constructions 5 and 2 combined using Theorem 8 and M2, where $T_2 = \exp(O(\log^3\log n)) = \mathsf{quasipoly}(\log n)$.
M9	$O(d^{g+3+\beta}\frac{T_3^\ell\log n}{(1-p)^{g+2}})$	$\Omega(pd^{2-\beta}\frac{\log n}{1-p})$	Exp: Constructions 5 and 2 combined using Theorem 9 and M2, where $\beta > 0$ is any arbitrary constant and $T_3 = ((\log n)(\log d))^{1+u/\beta} = \mathsf{poly}(\log n, \log d)$.
	$\Omega(d^{g+2}\log_d n + ed^{g+1})$	e	Lower bound (see Section 4).

this extended abstract we will mainly present the constructions and omit proof details that can be found in the full version of the paper.

1.1 Preliminaries

For a matrix M, we denote by $M[i, j]$ the entry of M at the ith row and jth column. Similarly we denote the ith entry of a vector v by $v(i)$. The *support* of a vector $x \in \{0, 1\}^n$, denoted by $\mathsf{supp}(x)$, is a subset of $[n] := \{1, \ldots, n\}$ such that $i \in \mathsf{supp}(x)$ iff $x(i) = 1$. The Hamming weight of x, denote by $\mathsf{wgt}(x)$ is defined as $|\mathsf{supp}(x)|$. For an $m \times n$ Boolean matrix M and $S \subseteq [n]$, we denote by $M|_S$ the $m \times |S|$ submatrix of M formed by restricting M to the columns picked by S. Moreover, for a vector $x \in \{0, 1\}^n$, we use $M[x]_{\ell,u}$ to denote the set of vectors in $\{0, 1\}^m$ that correctly encode the measurement outcomes resulting from non-adaptive threshold tests defined by the measurement matrix M on x using threshold parameters ℓ, u. In the gap-free case, this set may only have a single element that we denote by $M[x]_u$. Thus, for any $y \in M[x]_{\ell,u}$ we have $y(i) = 1$ if $|\mathsf{supp}(M_j) \cap \mathsf{supp}(x)| \geq u$, and $y(i) = 0$ if $|\mathsf{supp}(M_j) \cap \mathsf{supp}(x)| < \ell$, where M_j indicates the jth row of M.

The *min-entropy* of a distribution $\mathcal{X}$ with finite support Ω is given by $H_\infty(\mathcal{X}) := \min_{x\in\Omega}\{-\log \mathcal{X}(x)\}$, where $\mathcal{X}(x)$ is the probability that $\mathcal{X}$ assigns to the outcome x. The *statistical distance* between two distributions $\mathcal{X}$ and $\mathcal{Y}$ defined on the same finite space Ω is given by $\frac{1}{2}\sum_{s\in\Omega}|\mathcal{X}(s) - \mathcal{Y}(s)|$, which is half the ℓ_1 distance of the two distributions when regarded as vectors of probabilities over Ω. Two distributions $\mathcal{X}$ and $\mathcal{Y}$ are said to be ϵ-close if their statistical distance is at most ϵ. We will use the shorthand $\mathcal{U}_n$ for the uniform distribution on $\{0,1\}^n$, and $X \sim \mathcal{X}$ for a random variable X drawn from a distribution $\mathcal{X}$. The main technical tool that we use in our explicit constructions is the notion of *lossless condensers*. Formally, a function $f\colon \{0,1\}^{\tilde{n}} \times \{0,1\}^t \to \{0,1\}^{\tilde{\ell}}$ is a strong *lossless condenser* for entropy k and with *error* ϵ (in short, (k,ϵ)-condenser) if for every distribution $\mathcal{X}$ on $\{0,1\}^{\tilde{n}}$ with min-entropy at least k, random variable $X \sim \mathcal{X}$ and a *seed* $Y \sim \mathcal{U}_t$, the distribution of $(Y, f(X,Y))$ is ϵ-close to some distribution $(\mathcal{U}_t, \mathcal{Z})$ with min-entropy at least $t + k$. A condenser is *explicit* if it is polynomial-time computable in its input length.

2 Variations of Disjunct Matrices

The main combinatorial tool used by Chen and Fu in their non-adaptive scheme is the following generalization of the standard notion of disjunct matrices that we refer to as *strongly disjunct* matrices.

Definition 1. A matrix (with at least $d + u$ columns) is said to be strongly $(d, e; u)$-disjunct if for every choice of $d + u$ columns $C_1, \ldots, C_u, C'_1, \ldots, C'_d$, all distinct, we have $|\cap_{i=1}^{u} \mathsf{supp}(C_i) \setminus \cup_{i=1}^{d} \mathsf{supp}(C'_i)| > e$.

Observe that, $(d, e; u)$-disjunct matrices are, in particular, $(d', e'; u')$-disjunct for any $d' \le d$, $e' \le e$, and $u' \le u$. Moreover, *classical* (d, e)-disjunct matrices that are extensively used in group testing literature (see [2, Ch. 7]) correspond to the special case $u = 1$. We will refer to $e \ge 0$ as the *error parameter* of the matrix.

To make the main ideas more transparent, until Section 4 we will focus on the gap-free case where $\ell = u$. The extension to nonzero gaps is straightforward and will be discussed in Section 4. Moreover, often we will implicitly assume that the Hamming weight of the Boolean vector that is to be identified is at least u (since otherwise, any $(u-1)$-sparse vector would be confused with the all-zeros vector). Moreover, we will take the thresholds ℓ, u as fixed constants while the parameters d and n are allowed to grow.

The notion of strongly disjunct matrices, in its general form, has been studied in the literature under different names and equivalent formulations, e.g., superimposed (u, d)-designs/codes and (u, d) cover-free families. An important motivation for the study of this notion is the *hidden hypergraph-learning problem* (cf. [2, Ch. 12]). In this problem, one wishes to learn a "small" hidden subgraph of a given hypergraph by queries that each specify a set of vertices and output positive iff the subgraph induced by the specified set contains an edge from the hidden subgraph. It is known that [12,13], in the hypergraph-learning problem, any suitable grouping strategy defines a strongly disjunct matrix, and vice versa.

- *Given:* An $(\tilde{n}, k, \tilde{d})_q$ error-correcting code $\mathcal{C} \subseteq [q]^{\tilde{n}}$, and integer parameter $u > 0$.
- *Output:* An $m \times n$ Boolean matrix M, where $n = q^k$, and $m = \tilde{n}q^u$.
- *Construction:* First, consider the mapping $\varphi\colon [q] \to \{0,1\}^{q^u}$ from q-ary symbols to column vectors of length q^u defined as follows. Index the coordinates of the output vector by the u-tuples from the set $[q]^u$. Then $\varphi(x)$ has a 1 at position $(a_1, \ldots, a_u)$ iff there is an $i \in [u]$ such that $a_i = x$. Arrange all codewords of $\mathcal{C}$ as columns of an $\tilde{n} \times q^k$ matrix M' with entries from $[q]$. Then replace each entry x of M' with $\varphi(x)$ to obtain the output $m \times n$ matrix M.

Construction 1. Extension of Kautz-Singleton's method [9]

The key observation made by Chen and Fu [6] is that threshold group testing corresponds to the special case of the hypergraph learning problem where the hidden graph is known to be a clique. It follows that strongly $(d, e; u)$-disjunct matrices are suitable choices for the measurement matrices in threshold group testing with upper threshold u, sparsity d, and when up to $\lfloor e/2 \rfloor$ incorrect measurement outcomes are allowed.

Nonconstructively, a probabilistic argument akin to the standard argument for the case of classical disjunct matrices (see [2, Ch. 7]) can be used to show that strongly $(d, e; u)$-disjunct matrices exist with $m = O(d^{u+1}(\log(n/d))/(1-p)^2)$ rows and error parameter $e = \Omega(pd\log(n/d)/(1-p)^2)$, for any noise parameter $p \in [0, 1)$. On the negative side, however, several concrete lower bounds are known for the number of rows of such matrices [14,15,16]. In asymptotic terms, these results show that one must have $m = \Omega(d^{u+1}\log_d n + ed^u)$, and thus, the probabilistic upper bound is essentially optimal.

For the underlying strongly disjunct matrix, Chen and Fu [6] use a greedy construction [17] that achieves, for any $e \geq 0$, $O((e+1)d^{u+1}\log(n/d))$ rows, but may take exponential time in the size of the resulting matrix. Nevertheless, as observed by several researchers [15,18,12,13], a classical explicit construction of combinatorial designs due to Kautz and Singleton [9] can be extended to construct strongly disjunct matrices. This concatenation-based construction transforms any error-correcting code having large distance into a strongly disjunct matrix. The general transformation is given in Construction 1, and rows M2–M5 of Table 1 show the bounds achieved by this construction when various families of codes is used[2].

Even though, as discussed above, the general notion of strongly $(d, e; u)$-disjunct matrices is sufficient for threshold group testing with upper threshold u, in this section we show that a new, weaker, notion of disjunct matrices as defined below (which turns out to be *strictly* weaker when $u > 1$), would also suffice. We also define an "auxiliary" notion of *regular* matrices.

Definition 2. A Boolean matrix M with n columns is called $(d, e; u)$-regular if for every subset of columns $S \subseteq [n]$ (called the *critical set*) and every $Z \subseteq [n]$

[2] Some of these bounds are derived for the first time in this work, and will be elaborated upon in the full version of the paper.

(called the *zero set*) such that $u \leq |S| \leq d$, $|Z| \leq |S|$, $S \cap Z = \emptyset$, there are more than e rows of M at which $M|_S$ has weight exactly u and (at the same rows) $M|_Z$ has weight zero. Any such row is said to *u-satisfy* S and Z. If, in addition, for every *distinguished column* $i \in S$, more than e rows of M both u-satisfy S and Z and have a 1 at the ith column, the matrix is called $(d, e; u)$-disjunct (and the corresponding "good" rows are said to u-satisfy i, S, and Z).

It is easy to verify that (assuming $2d \leq n$) the classical notion of $(2d-1, e)$-disjunct matrices is equivalent to strongly $(2d-1, e; 1)$-disjunct and $(d, e; 1)$-disjunct. Moreover, any $(d, e; u)$-disjunct matrix is $(d, e; u)$-regular, $(d-1, e; u-1)$-regular, and (d, e)-disjunct (but the reverse implications do not in general hold). Therefore, the above-mentioned lower bound of $m = \Omega(d^2 \log_d n + ed)$ that applies for (d, e)-disjunct matrices holds for $(d, e; u)$-disjunct matrices as well. The lemma below shows that our notion of disjunct matrices is necessary and sufficient for the purpose of threshold group testing:

Lemma 3. *Let M be an $m \times n$ Boolean matrix that is $(d, e; u)$-disjunct. Then for every distinct d-sparse vectors $x, x' \in \{0,1\}^n$ such that*[3] $\mathsf{supp}(x) \nsubseteq \mathsf{supp}(x')$, $\mathsf{wgt}(x) \geq |\mathsf{supp}(x') \setminus \mathsf{supp}(x)|$ *and* $\mathsf{wgt}(x) \geq u$, *we have*

$$|\mathsf{supp}(M[x]_u) \setminus \mathsf{supp}(M[x']_u)| > e. \tag{1}$$

Conversely, assuming $d \geq 2u$, if M satisfies (1) for every choice of x and x' as above, it must be $(\lfloor d/2 \rfloor, e; u)$-disjunct.

- *Given:* Boolean matrices M_1 and M_2 that are $m_1 \times n$ and $m_2 \times n$, respectively.
- *Output:* An $m \times n$ Boolean matrix $M_1 \odot M_2$, where $m := m_1 m_2$.
- *Construction:* Let the rows of $M := M_1 \odot M_2$ be indexed by the set $[m_1] \times [m_2]$. Then the row corresponding to (i, j) is defined as the bit-wise or of the ith row of M_1 and the jth row of M_2.

Construction 2. Direct product of measurement matrices

We will use regular matrices as building blocks in our constructions of disjunct matrices to follow. The connection with disjunct matrices is made apparent through a direct product of matrices defined in Construction 2. Intuitively, using this product, regular matrices can be used to transform any measurement matrix suitable for the standard group testing model to one with comparable properties in the threshold model. The following lemma formalizes this idea.

Lemma 4. *Let M_1 and M_2 be Boolean matrices with n columns, such that M_1 is $(d-1, e_1; u-1)$-regular. Let $M := M_1 \odot M_2$, and suppose that for d-sparse Boolean vectors $x, x' \in \{0,1\}^n$ such that $\mathsf{wgt}(x) \geq \mathsf{wgt}(x')$, we have $|\mathsf{supp}(M_2[x]_1) \setminus \mathsf{supp}(M_2[x']_1)| \geq e_2$. Then, $|\mathsf{supp}(M[x]_u) \setminus \mathsf{supp}(M[x']_u)| \geq (e_1 + 1)e_2$.*

[3] Note that at least one of the two possible orderings of any two distinct d-sparse vectors, at least one having weight u or more, satisfies this condition.

As a corollary it follows that, when M_1 is a $(d-1, e_1; u-1)$-regular and M_2 is a (d, e_2)-disjunct matrix, the product $M := M_1 \odot M_2$ will distinguish between any two distinct d-sparse vectors (of weight at least u) in at least $(e_1+1)(e_2+1)$ positions of the measurement outcomes. This combined with Lemma 3 would imply that M is, in particular, $(\lfloor d/2 \rfloor, (e_1+1)(e_2+1)-1; u)$-disjunct. However, using a direct argument similar to the above lemma it is possible to obtain a slightly better result, given by Lemma 5.

Lemma 5. *Suppose that M_1 is a $(d, e_1; u-1)$-regular and M_2 is a $(2d, e_2)$-disjunct matrix. Then $M_1 \odot M_2$ is a $(d, (e_1+1)(e_2+1)-1; u)$-disjunct matrix.*

As another particular example, we remark that measurement matrices constructed in [19] that are not necessarily disjunct but allow approximation of sparse vectors in highly noisy settings of the standard group testing model (as well as those used in adaptive two-stage schemes; cf. [20] and the references therein), can be combined with regular matrices to offer the same qualities in the threshold model. In the same way, numerous existing results in group testing can be ported to the threshold model by using Lemma 4.

3 Constructions

In this section, we introduce several constructions of regular and disjunct matrices. Our first construction, described in Construction 3, is a randomness-efficient probabilistic construction that can be analyzed using standard techniques from the probabilistic method. The bounds obtained by this construction are given by Lemma 6 below. The amount of random bits required by this construction is polynomially bounded in d and $\log n$, which is significantly smaller than it would be had we picked the entries of M fully independently.

- *Given:* Integer parameters n, m', d, u.
- *Output:* An $m \times n$ Boolean matrix M, where $m := m'\lceil \log(d/u) \rceil$.
- *Construction:* Let $r := \lceil \log(d/u) \rceil$. Index the rows of M by $[r] \times [m']$. Sample the (i,j)th row of M independently from a $(u+1)$-wise independent distribution on n bit vectors, where each individual bit has probability $1/(2^{i+2}u)$ of being 1.

Construction 3. Probabilistic construction of regular and disjunct matrices

Lemma 6. *For every $p \in [0,1)$ and integer parameter $u > 0$, Construction 3 with[4] $m' = O_u(d \log(n/d)/(1-p)^2)$ (resp., $m' = O_u(d^2 \log(n/d)/(1-p)^2)$) outputs a $(d, \Omega_u(pm'); u)$-regular (resp., $(d, \Omega_u(pm'/d); u)$-disjunct) matrix with probability $1 - o(1)$.*

[4] The subscript in $O_u(\cdot)$ and $\Omega_u(\cdot)$ implies that the hidden constant in the asymptotic notation is allowed to depend on u.

- *Given:* A strong lossless (k, ϵ)-condenser $f\colon \{0,1\}^{\tilde{n}} \times \{0,1\}^t \to \{0,1\}^{\tilde{\ell}}$, integer parameter $u \geq 1$ and real parameter $p \in [0,1)$ such that $\epsilon < (1-p)/16$,
- *Output:* An $m \times n$ Boolean matrix M, where $n := 2^{\tilde{n}}$ and $m = 2^{t+k} O_u(2^{u(\tilde{\ell}-k)})$.
- *Construction:* Let $G_1 = (\{0,1\}^{\tilde{\ell}}, \{0,1\}^k, E_1)$ be any bipartite bi-regular graph with left vertex set $\{0,1\}^{\tilde{\ell}}$, right vertex set $\{0,1\}^k$, left degree $d_\ell := 8u$, and right degree $d_r := 8u2^{\tilde{\ell}-k}$. Replace each right vertex v of G_1 with $\binom{d_r}{u}$ vertices, one for each subset of size u of the vertices on the neighborhood of v, and connect them to the corresponding subsets. Denote the resulting graph by $G_2 = (\{0,1\}^{\tilde{\ell}}, V_2, E_2)$, where $|V_2| = 2^k \binom{d_r}{u}$. Define the bipartite graph $G_3 = (\{0,1\}^n, V_3, E_3)$, where $V_3 := \{0,1\}^t \times V_2$, as follows: Each left vertex $x \in \{0,1\}^n$ is connected to $(y, \Gamma_2(f(x,y)))$, for each $y \in \{0,1\}^t$, where $\Gamma_2(\cdot)$ denotes the neighborhood function of G_2 (i.e., $\Gamma_2(v)$ denotes the set of vertices adjacent to v in G_2). The output matrix M is the bipartite adjacency matrix of G_3.

Construction 4. A building block for construction of regular matrices

- *Given:* Integer parameters $d \geq u \geq 1$, real parameter $p \in [0,1)$, and a family $f_0, \ldots, f_r$ of strong lossless condensers, where $r := \lceil \log(d/u') \rceil$ and u' is the smallest power of two such that $u' \geq u$. Each $f_i\colon \{0,1\}^{\tilde{n}} \times \{0,1\}^t \to \{0,1\}^{\tilde{\ell}(i)}$ is assumed to be a strong lossless $(k(i), \epsilon)$-condenser, where $k(i) := \log u' + i + 1$ and $\epsilon < (1-p)/16$.
- *Output:* An $m \times n$ Boolean matrix M, where $n := 2^{\tilde{n}}$ and $m = 2^t d \sum_{i=0}^{r} O_u(2^{u(\tilde{\ell}(i)-k(i))})$.
- *Construction:* For each $i \in \{0, \ldots, r\}$, denote by M_i the output matrix of Construction 4 when instantiated with f_i as the underlying condenser, and by m_i its number of rows. Define $r_i := 2^{r-i}$ and let M'_i denote the matrix obtained from M_i by repeating each row r_i times. Construct the output matrix M by stacking $M'_0, \ldots, M'_r$ on top of one another.

Construction 5. Regular matrices from strong lossless condensers

One of the technical contributions of this paper is a construction of regular matrices using strong lossless condensers. Details of the construction are described in Construction 5 that assumes a family of lossless condensers with different entropy requirements[5], and in turn, uses Construction 4 as a building block. The following theorem analyzes the obtained parameters without specifying any particular choice for the underlying family of condensers.

Theorem 7. *The $m \times n$ matrix M output by Construction 5 is $(d, p\gamma 2^t; u)$-regular, where $\gamma = \max\{1, \Omega_u(d \cdot \min\{2^{k(i)-\tilde{\ell}(i)} : i = 0, \ldots, r\})\}$.*

[5] We have assumed that all the functions in the family have the same seed length t. If this is not the case, one can trivially set t to be the largest seed length in the family.

3.1 Instantiations

We now discuss instantiations the result obtained in Theorem 7 by various choices of the family of lossless condensers. The crucial factors that influence the number of measurements are the seed length and the output length of the condenser.

Nonconstructively, it can be shown that strong (k, ϵ) lossless condensers with input length $\tilde{n}$, seed length $t = \log \tilde{n} + \log(1/\epsilon) + O(1)$, and output length $\tilde{\ell} = k + \log(1/\epsilon) + O(1)$ exist, and moreover, almost matching lower bounds are known [7]. In fact, the optimal parameters can be achieved by a random function with overwhelming probability. In this work, we consider two important explicit constructions of lossless condensers. Namely, one based on "zig-zag products" due to Capalbo et al. [7] and another, coding theoretic, construction due to Guruswami et al. [8].

Theorem 8. *[7] For every $k \leq \tilde{n} \in \mathbb{N}$, $\epsilon > 0$ there is an explicit lossless (k, ϵ) condenser with seed length $O(\log^3(\tilde{n}/\epsilon))$ and output length $k + \log(1/\epsilon) + O(1)$.*

Theorem 9. *[8] For all constants $\alpha \in (0, 1)$ and every $k \leq \tilde{n} \in \mathbb{N}$, $\epsilon > 0$ there is an explicit strong lossless (k, ϵ) condenser with seed length $t = (1 + 1/\alpha)\log(\tilde{n}k/\epsilon) + O(1)$ and output length $\tilde{\ell} = t + (1 + \alpha)k$.*

As a result, we use Theorem 7 with the above condensers to obtain the following.

Theorem 10. *Let $u > 0$ be fixed, and $p \in [0, 1)$ be a real parameter. Then for integer parameters $d, n \in \mathbb{N}$ where $u \leq d \leq n$,*

1. *Using an optimal lossless condenser in Construction 5 results in an $m_1 \times n$ matrix M_1 that is $(d, e_1; u)$-regular, where $m_1 = O(d(\log n)(\log d)/(1-p)^{u+1})$ and $e_1 = \Omega(pd \log n)$,*
2. *Using the lossless condenser of Theorem 8 in Construction 5 results in an $m_2 \times n$ matrix M_2 that is $(d, e_2; u)$-regular, where $m_2 = O(T_2 d(\log d)/(1-p)^u)$ for some $T_2 = \exp(O(\log^3((\log n)/(1-p)))) = \mathsf{quasipoly}(\log n)$, and $e_2 = \Omega(pdT_2(1-p))$.*
3. *Let $\beta > 0$ be any fixed constant. Then Construction 5 can be instantiated using the lossless condenser of Theorem 9 so that we obtain an $m_3 \times n$ matrix M_3 that is $(d, e_3; u)$-regular, where $m_3 = O(T_3^{1+u} d^{1+\beta}(\log d))$ for the quantity T_3 defined as $T_3 := ((\log n)(\log d)/(1-p))^{1+u/\beta} = \mathsf{poly}(\log n, \log d)$, and $e_3 = \Omega(p \max\{T_3, d^{1-\beta/u}\})$.*

Finally, by combining this result with Lemma 5 using any explicit construction of classical disjunct matrices, we will obtain $(d, e; u)$-disjunct matrices that can be used in the threshold model with any fixed threshold, sparsity d, and error tolerance $\lfloor e/2 \rfloor$. In particular, using the coding-theoretic explicit construction of nearly optimal classical disjunct matrices [10] (a special case of what shown in row M2 of Table 1), we obtain $(d, e; u)$-disjunct matrices with $m = O(m'd^2(\log n)/(1-p)^2)$ rows and error parameter $e = \Omega(e'pd(\log n)/(1-p))$, where m' and e' are respectively the number of rows and error parameter of

any of the regular matrices obtained in Theorem 10. We note that in all cases, the final dependence on the sparsity parameter d is, roughly, $O(d^3)$ which has an exponent independent of the threshold u. Rows M7–M9 of Table 1 summarize the obtained parameters for the general case (with arbitrary gaps). We see that, when d is not negligibly small (e.g., $d = n^{1/10}$), the bounds obtained by our explicit constructions are significantly better than those offered by strongly disjunct matrices.

4 The Case with Positive Gaps

In preceding sections we have focused on the case where $g = 0$. However, we observe that all the techniques that we have developed in this work can be extended to the positive-gap case by the following observations:

1. Definition 2 can be adapted to allow more than one distinguished columns in disjunct matrices. In particular, in general we may require the matrix M to have more than e rows that u-satisfy every choice of a critical set S, a zero set Z, and any $g+1$ designated columns $D \subseteq S$ (at which all entries of the corresponding rows must be 1). Denote this generalized notion by $(d, e; u, g)$-disjunct matrices. It is straightforward to extend Lemma 3 to show that the notion of $(d, e; u, g)$-disjunct matrices is necessary and sufficient to capture non-adaptive threshold group testing with upper threshold u and gap g.
2. Lemma 6 can be generalized to show that Construction 3 (with probability $1-o(1)$) results in a $(d, \Omega_u(pd\log(n/d)/(1-p)^2); u, g)$-disjunct matrix if the number of measurements is increased by a factor $O(d^g)$.
3. Lemma 4 can be extended to positive gaps, by taking M_1 as a $(d-1, e_1; \ell-1)$-regular matrix, provided that, for every $y \in M_2[x]_{1,g+1}$ and $y' \in M_2[x']_{1,g+1}$, we have $|\mathsf{supp}(y) \setminus \mathsf{supp}(y')| \geq e_2$. In particular this is the case if M_2 is strongly $(d, e_2 - 1; g+1)$-disjunct[6]. Similarly for Lemma 5, M_2 must be taken as a strongly $(2d, e_2; g+1)$-disjunct matrix. Consequently, using the coding-theoretic construction of strongly disjunct matrices introduced in Construction 1, our explicit constructions of $(d, e; u)$-disjunct matrices can be extended to the gap model at the cost of a factor $O(d^g)$ increase in the number of measurements (as summarized in Table 1).
4. Observe that a $(d, e; u, g)$-disjunct matrix is in particular, strongly $(d-g, e; g+1)$-disjunct and thus, the lower bound $\Omega(d^{g+2}\log_d n + ed^{g+1})$ on the number of rows of strongly disjunct matrices applies to them as well.

5 Concluding Remarks

In this work we have introduced the combinatorial notion of $(d, e; u)$-regular matrices, that is used as an intermediate tool towards obtaining threshold testing designs. Even though our construction, assuming an optimal lossless condenser, matches the probabilistic upper bound for regular matrices, the number

[6] Here we are also considering the unavoidable assumption that $\max\{|\mathsf{supp}(x) \setminus \mathsf{supp}(x')|, |\mathsf{supp}(x') \setminus \mathsf{supp}(x)|\} > g$.

of measurements in the resulting threshold testing scheme will be larger than the probabilistic upper bound by a factor of $\Omega(d \log n)$. Thus, an outstanding question is coming up with a direct construction of disjunct matrices that match the probabilistic upper bound. Moreover, we have assumed the threshold u to be a fixed constant, allowing the constants hidden in asymptotic notions to have a poor dependence on u. An outstanding question is whether the number of measurements can be reasonably controlled when u becomes large; e.g., $u = \Omega(d)$. Another interesting problem is decoding. While our constructions can combinatorially guarantee identification of sparse vectors, for applications it is important to have an efficient reconstruction algorithm as well. Contrary to the case of strongly disjunct matrices that allow a straightforward decoding procedure (cf. [6]), it is not clear whether in general our notion of disjunct matrices allow efficient decoding, and thus it becomes important to look for constructions that are equipped with efficient reconstruction algorithms.

References

1. Dorfman, R.: The detection of defective members of large populations. Annals of Mathematical Statistics 14, 436–440 (1943)
2. Du, D.Z., Hwang, F.K.: Combinatorial Group Testing and its Applications, 2nd edn. World Scientific, Singapore (2000)
3. Du, D.Z., Hwang, F.K.: Pooling Designs and Nonadaptive Group Testing. World Scientific, Singapore (2006)
4. Damaschke, P.: Threshold group testing. In: Ahlswede, R., Bäumer, L., Cai, N., Aydinian, H., Blinovsky, V., Deppe, C., Mashurian, H. (eds.) General Theory of Information Transfer and Combinatorics. LNCS, vol. 4123, pp. 707–718. Springer, Heidelberg (2006)
5. Chang, H., Chen, H.B., Fu, H.L., Shi, C.H.: Reconstruction of hidden graphs and threshold group testing. Journal of Combinatorial Optimization (2010)
6. Chen, H.B., Fu, H.L.: Nonadaptive algorithms for threshold group testing. Discrete Applied Mathematics 157, 1581–1585 (2009)
7. Capalbo, M., Reingold, O., Vadhan, S., Wigderson, A.: Randomness conductors and constant-degree expansion beyond the degree/2 barrier. In: Proceedings of the 34th STOC, pp. 659–668 (2002)
8. Guruswami, V., Umans, C., Vadhan, S.: Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. In: Proc. of the 22nd IEEE CCC (2007)
9. Kautz, W., Singleton, R.: Nonrandom binary superimposed codes. IEEE Transactions on Information Theory 10, 363–377 (1964)
10. Porat, E., Rothschild, A.: Explicit non-adaptive combinatorial group testing schemes. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 748–759. Springer, Heidelberg (2008)
11. Tsfasman, M.A., Vlăduţ, S.G., Zink, T.: Modular curves, Shimura curves, and Goppa codes better than the Varshamov-Gilbert bound. Math. Nachrichten 109, 21–28 (1982)
12. Gao, H., Hwang, F.K., Thai, M.T., Wu, W., Znati, T.: Construction of $d(H)$-disjunct matrix for group testing in hypergraphs. Journal of Combinatorial Optimization 12, 297–301 (2006)

13. Chen, H.B., Du, D.Z., Hwang, F.K.: An unexpected meeting of four seemingly unrelated problems: graph testing, DNA complex screening, superimposed codes and secure key distribution. J. Comb. Opt. 14(2-3), 121–129 (2007)
14. Stinson, D., Wei, R., Zhu, L.: Some new bounds for cover-free families. Journal of Combinatorial Theory, Series A 90, 224–234 (2000)
15. D'yachkov, A., Vilenkin, P., Macula, A., Torney, D.: Families of finite sets in which no intersection of ℓ sets is covered by the union of s others. Journal of Combinatorial Theory, Series A 99, 195–218 (2002)
16. Stinson, D., Wei, R.: Generalized cover-free families. Discrete Mathematics 279, 463–477 (2004)
17. Chen, H.B., Fu, H.L., Hwang, F.K.: An upper bound of the number of tests in pooling designs for the error-tolerant complex model. Optimization Letters 2(3), 425–431 (2008)
18. Kim, H.K., Lebedev, V.: On optimal superimposed codes. Journal of Combinatorial Designs 12, 79–91 (2004)
19. Cheraghchi, M.: Noise-resilient group testing: Limitations and constructions. In: Gębala, M. (ed.) FCT 2009. LNCS, vol. 5699, pp. 62–73. Springer, Heidelberg (2009)
20. Cheng, Y., Du, D.Z.: New constructions of one- and two-stage pooling designs. Journal of Computational Biology 15(2), 195–205 (2008)

Testing Non-uniform k-Wise Independent Distributions over Product Spaces (Extended Abstract)*

Ronitt Rubinfeld** and Ning Xie***

MIT and Tel Aviv University
MIT
{ronitt,ningxie}@csail.mit.edu

Abstract. A distribution D over $\Sigma_1 \times \cdots \times \Sigma_n$ is called (non-uniform) k-wise independent if for any set of k indices $\{i_1, \ldots, i_k\}$ and for any $z_1 \cdots z_k \in \Sigma_{i_1} \times \cdots \times \Sigma_{i_k}$, $\Pr_{X \sim D}[X_{i_1} \cdots X_{i_k} = z_1 \cdots z_k] = \Pr_{X \sim D}[X_{i_1} = z_1] \cdots \Pr_{X \sim D}[X_{i_k} = z_k]$. We study the problem of testing (non-uniform) k-wise independent distributions over product spaces. For the uniform case we show an upper bound on the distance between a distribution D from the set of k-wise independent distributions in terms of the sum of Fourier coefficients of D at vectors of weight at most k. Such a bound was previously known only for the binary field. For the non-uniform case, we give a new characterization of distributions being k-wise independent and further show that such a characterization is robust. These greatly generalize the results of Alon et al. [1] on uniform k-wise independence over the binary field to non-uniform k-wise independence over product spaces. Our results yield natural testing algorithms for k-wise independence with time and sample complexity sublinear in terms of the support size when k is a constant. The main technical tools employed include discrete Fourier transforms and the theory of linear systems of congruences.

1 Introduction

Nowadays we are both blessed and cursed by the colossal amount of data available for processing. In many situations, simply scanning the whole data set once can be a daunting task. It is then natural to ask what we can do in *sublinear time*. For many computational questions, if instead of asking the decision version of the problems, one can relax the questions and consider the analogous property testing problems, then sublinear algorithms are often possible. See survey articles [18,35,27,14].

* A full version of this paper is available at http://people.csail.mit.edu/ningxie/papers/RX09.pdf

** Research supported by NSF grants 0514771, 0728645,0732334, Marie-Curie International Reintegration Grant PIRG03-GA-2008-231077, and Israel Science Foundation Grant Numbers 1147/09 and 1675/09.

*** Research supported in part by an Akamai Presidential Fellowship and NSF grants 0514771, 0728645 and 0732334.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 565–581, 2010.

Property testing algorithms [36,19] are usually based on *robust characterizations* of the objects being tested. For instance, the linearity test introduced in [12] is based on the characterization that a function is linear if and only if the linearity test (for all x and y, it holds that $f(x) + f(y) = f(x + y)$) has acceptance probability 1. Moreover, the characterization is robust in the sense that if the linearity test accepts a function with probability close to 1, then the function must be also close to some linear function. Property testing often leads to a new understanding of well-studied problems and sheds insight on related problems.

In this work, we show robust characterizations of k-wise independent distributions over discrete product spaces and give sublinear-time testing algorithms based on these robust characterizations. Note that distributions over product spaces are in general not *product distributions*, which by definition are n-wise independent distributions.

The k-wise Independent Distributions: For finite set Σ, a discrete probability distribution D over Σ^n is (non-uniform) *k-wise independent* if for any set of k indices $\{i_1, \ldots, i_k\}$ and for all $z_1, \ldots, z_k \in \Sigma$, $\Pr_{X\sim D}[X_{i_1} \cdots X_{i_k} = z_1 \cdots z_k] = \Pr_{X\sim D}[X_{i_1} = z_1] \cdots \Pr_{X\sim D}[X_{i_k} = z_k]$. That is, restricting D to any k coordinates gives rise to a fully independent distribution. For the special case of $\Pr_{X\sim D}[X_i = z] = \frac{1}{|\Sigma|}$ for all i and all $z \in \Sigma$, we refer to the distribution as *uniform* k-wise independent[1]. A distribution is *almost k-wise independent* if its restriction to any k coordinates is very close to some independent distribution. k-wise independent distributions look independent "locally" to any observer of only k coordinates, even though they may be far from fully independent "globally". Furthermore, k-wise independent distributions can be constructed with exponentially smaller support sizes than fully independent distributions. Because of these useful properties, k-wise independent distributions have many applications in both probability theory and computational complexity theory [23,25,28,31].

Given samples drawn from a distribution, it is natural to ask, how many samples are required to tell whether the distribution is k-wise independent or far from k-wise independent, where by "far from k-wise independent" we mean that the distribution has large statistical distance from *any* k-wise independent distribution. Usually the time and query complexity of distribution testing algorithms are measured against the support size of the distributions; For example, algorithms that test distributions over $\{0,1\}^n$ with time complexity $o(2^n)$ are said to be sublinear-time testing algorithms.

Alon, Goldreich and Mansour [4] implicitly give the first robust characterization of k-wise independence. Alon et al. [1] improve the bounds in [4] and also give efficient testing algorithms. All of these results consider only uniform distributions over GF(2). Our work generalizes previous results in two ways: to distributions over arbitrary finite product spaces and to non-uniform k-wise independent distributions.

[1] In literature the term "k-wise independence" usually refers to uniform k-wise independence.

1.1 Our Results

Let $\Sigma = \{0, 1, \ldots, q-1\}$ be the alphabet and let $D : \Sigma^n \to [0, 1]$ be the distribution to be tested. For any vector $\boldsymbol{a} \in \Sigma^n$, the Fourier coefficient of distribution D at $\boldsymbol{a}$ is $\hat{D}(\boldsymbol{a}) = \sum_{\boldsymbol{x} \in \Sigma^n} D(\boldsymbol{x}) e^{\frac{2\pi i}{q} \sum_{j=1}^{n} a_j x_j} = \mathbf{E}_{\boldsymbol{x} \sim D} \left[e^{\frac{2\pi i}{q} \sum_{j=1}^{n} a_j x_j} \right]$. The *weight* of $\boldsymbol{a}$ is the number of non-zero entries in $\boldsymbol{a}$. It is a folklore fact that a distribution D is uniform k-wise independent if and only if for all non-zero vectors $\boldsymbol{a}$ of weight at most k, $\hat{D}(\boldsymbol{a}) = 0$. A natural test for k-wise independence is thus the following *Generic Algorithm* for testing k-wise independence shown in Fig. 1.

Generic Algorithm for Testing Uniform k-wise Independence

1. Sample D uniformly and independently M times
2. Use these samples to estimate all the low-weight Fourier coefficients
3. **Accept** if the magnitudes of *all* the estimated Fourier coefficients are at most δ

Fig. 1. A generic algorithm for testing uniform k-wise independence

However, in order to prove that the generic algorithm works, one needs to show that the simple characterization of k-wise independence is *robust* in the sense that, for any distribution D, if all its Fourier coefficients at vectors of weight at most k are at most δ (in magnitude), then D is $\epsilon(\delta)$-close to some uniform k-wise independent distribution, where the closeness parameter ϵ depends, among other things, on the error parameter δ.[2] Furthermore, the query and time complexity of the generic testing algorithm will depend on the underlying upper bound. One of our main results is the following robust characterization of uniform k-wise independence. Let $\Delta(D, D_{\text{kwi}})$ denote the distance between D and the set of k-wise independent distributions over $\{0, 1, \ldots, q-1\}^n$, then

$$\Delta(D, D_{\text{kwi}}) \leq \sum_{0 < \text{wt}(\boldsymbol{a}) \leq k} |\hat{D}(\boldsymbol{a})|.$$

Consequently, the sample complexity of our testing algorithm is $\tilde{O}(\frac{n^{2k}(q-1)^{2k}q^2}{\epsilon^2})$ and the time complexity is $\tilde{O}(\frac{n^{3k}(q-1)^{3k}q^2}{\epsilon^2})$, which are both sublinear when $k = O(1)$ and $q \leq \text{poly}(n)$. We further generalize these results to non-uniform k-wise independent distributions over product space, i.e., distributions over $\Sigma_1 \times \cdots \times \Sigma_n$, where $\Sigma_1, \ldots, \Sigma_n$ are finite sets.

We remark that another related problem, namely testing *almost k-wise independence* over product spaces admits a straightforward generalization of the testing algorithm in [1], which is shown there to work only for the (uniform) binary case. We refer interested readers to the full version of the paper.

[2] Note that, for *almost k-wise independence*, all the Fourier coefficients at vectors of weight at most k being small already implies that the distribution is almost k-wise independent.

Our results add a new understanding of the structures underlying (non-uniform) k-wise independent distributions and it is hoped that one may find other applications of these robust characterizations.

As is often the case, commutative rings demonstrate different algebraic structures from those of prime fields. For example, the recent improved construction [16] of 3-query locally decodable codes of Yekhanin [41] relies crucially on a set system construction [21] which holds only modulo composite numbers. Generalizing results in the binary field (or prime fields) to commutative rings often poses new technical challenges and requires additional new ideas. We hope our results may find future applications in generalizing other results working in the Boolean domains to general domains.

1.2 Techniques

Previous Techniques: Given a distribution D over binary field, a k-wise independent distribution is constructed in [4] by mixing D with a series of carefully chosen distributions to the given distribution in order to zero-out all the Fourier coefficients over subsets of size at most k. For a given subset S, the added distribution U_S is chosen such that, on the one hand it corrects the Fourier coefficient over S; on the other hand, U_S's Fourier coefficient of D over *any* other subset is zero. Using the orthogonality property of Hadamard matrices, they choose U_S to be the uniform distribution over all strings whose parity over S is 1 (or -1, depending on the sign of the distribution's bias over S). Although one can generalize it to work for prime fields, this construction breaks down when the alphabet size is a composite number.

For binary field a better bound is obtained in [1]. This is achieved by first working in the Fourier domain to remove all the first k-level Fourier coefficients of the input distribution. Such an operation ends up with a so-called "pseudo-distribution", since at some points the resulting function may assume negative values. Then a series of carefully chosen k-wise independent distributions are added to the pseudo-distribution to fix the negative points. This approach does not admit a direct generalization to the non-Boolean cases because, for larger domains, the pseudo-distributions are in general complex-valued. Nevertheless[3], one may use generalized Fourier expansion of real-valued functions to overcome this difficulty. We present this approach in the appendices of the full version of the paper. However, the bound obtained from this approach is weaker than our main results for the uniform case which we discuss shortly. Moreover, the proof is "non-constructive" in the sense that we are not aware of what distributions should we mix with the input distribution to make it a k-wise independent one. This drawback seems make it hard to generalize the approach to handle the non-uniform case. In contrast, our results on non-uniform k-wise independence relies crucially on the fact that the correction process for the uniform case is explicit and all the distributions used for mixing are of some special structure.

Uniform Distributions: Our results on uniform k-wise independent distributions extend the framework in [4]. As noted before, the key property used to mend a

[3] We thank an anonymous referee for pointing this out.

distribution into k-wise independent is the *orthogonality* relation between any pair of vectors. We first observe that all prime fields also enjoy this nice feature after some slight modifications. More specifically, for any two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ in $\mathbb{Z}_p^n$ that are *linearly independent*, the set of strings with $\sum_{i=1}^n a_i x_i \equiv j \pmod{\text{p}}$ are *uniformly* distributed over $S_{\boldsymbol{b},\ell} := \{\boldsymbol{x} : \sum_{i=1}^n b_i x_i \equiv \ell \pmod{\text{p}}\}$ for every $0 \leq \ell \leq p-1$. We will call this the *strong orthogonality* between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. The case when $q = |\Sigma|$ is not a prime is less straightforward. The main difficulty is that the strong orthogonality between pairs of vectors no longer holds, even when they are linearly independent. Suppose we wish to use some distribution $U_{\boldsymbol{a}}$ to correct the bias over $\boldsymbol{a}$. A simple but important observation is that we only need that $U_{\boldsymbol{a}}$'s Fourier coefficient at $\boldsymbol{b}$ to be zero, which is a much weaker requirement than the property of being strongly orthogonal between $\boldsymbol{a}$ and $\boldsymbol{b}$. Using a classical result in linear systems of congruences due to Smith [38], we are able to show that, when $\boldsymbol{a}$ satisfies $\gcd(a_1, \ldots, a_n) = 1$ and $\boldsymbol{b}$ is not a multiple of $\boldsymbol{a}$, the set of strings with $\sum_{i=1}^n a_i x_i \equiv j \pmod{\text{p}}$ are *uniformly* distributed over $S_{\boldsymbol{b},\ell}$ for ℓ's that lie in a *subgroup* of $\mathbb{Z}_q$ (compared with uniform distribution over the whole group $\mathbb{Z}_p$ for prime fields case). We refer to this as *weak orthogonality* between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. To zero-out the Fourier coefficients at $\boldsymbol{a}$, we instead bundle the Fourier coefficient at $\boldsymbol{a}$ with the Fourier coefficients at $\ell\boldsymbol{a}$ for every $\ell = 2, \ldots, q-1$, and treat them as Fourier coefficients defined in one-dimensional space with ℓ as the variable. This allows us to upper bound the total weights required to simultaneously correct *all* the Fourier coefficients at $\boldsymbol{a}$ and its multiples using only $U_{\boldsymbol{a}}$. We also generalize the result to product spaces of different alphabet sizes $\mathfrak{D} = \Sigma_1 \times \cdots \times \Sigma_n$.

Non-uniform Distributions: One possible way of extending the upper bounds for the uniform case to the non-uniform case would be to map non-uniform probabilities to uniform probabilities over a larger domain. For example, consider when $q = 2$ a distribution D with $\Pr_D[x_i = 0] = 0.501$ and $\Pr_D[x_i = 1] = 0.499$. We could map $x_i = 0$ and $x_i = 1$ uniformly to $\{1, \ldots, 501\}$ and $\{502, \ldots, 1000\}$, respectively and test if the transformed distribution D' over $\{1, \ldots, 1000\}$ is k-wise independent. Unfortunately, this approach produces a huge overhead on the distance upper bound, due to the fact that the alphabet size increases depends on the closeness of marginal probabilities over different symbols. However, in the previous example we would expect D behaves very much like the uniform case rather than with an additional factor of 1000 blowup in the alphabet size. Instead we take the following approach. Consider a stretching/compressing factor for each marginal probability $\Pr_D[x_i = z_j]$, where $z_j \in \Sigma$. Specifically, define $\theta_i(z_j) = \frac{1}{q \Pr_D[x_i = z_j]}$ so that $\theta_i(z_j) \Pr_D[x_i = z_j] = \frac{1}{q}$, the probability in the uniform distribution. If we multiply $D(\boldsymbol{x})$ for each $\boldsymbol{x}$ in the domain by the product of all n of these factors, the non-uniform k-wise independent distribution will be transformed into a uniform one. The hope is that distributions close to non-uniform k-wise independent will also be transformed into distributions that are close to uniform k-wise independent. However, this could give rise to exponentially large distribution weights at some points in the domain, making the task of estimating the corresponding Fourier coefficients intractable. Observe that,

intuitively for testing k-wise independence purposes, all we need to know are the "local" weight distributions. To be more specific, for a vector $\boldsymbol{a} \in \Sigma^n$, define the support of $\boldsymbol{a}$ by $\text{supp}(\boldsymbol{a}) = \{i \in [n] : a_i \neq 0\}$. For every non-zero vector $\boldsymbol{a}$ of weight at most k, we define a new *non-uniform Fourier coefficient* at $\boldsymbol{a}$ by first project D to $\text{supp}(\boldsymbol{a})$, then apply the stretching/compressing transformation and finally compute the Fourier coefficient based on the "transformed" local distribution. We are able to show a new characterization that D is a non-uniform k-wise independent distribution *if and only if* all these low-degree non-uniform Fourier coefficients are zero. This enable us to apply the Fourier coefficient correcting approach developed for the uniform cases. Roughly speaking, for any vector $\boldsymbol{a}$, we can find a (small-weight) distribution $U_{\boldsymbol{a}}$ such that mixing D with $U_{\boldsymbol{a}}$ zeroes-out the non-uniform Fourier coefficient at $\boldsymbol{a}$. But this $U_{\boldsymbol{a}}$ is the distribution to mix in the "transformed" world. We therefore apply some appropriate *inverse* stretching/compressing transformations to $U_{\boldsymbol{a}}$ to get $\tilde{U}_{\boldsymbol{a}}$, and show that mixing $\tilde{U}_{\boldsymbol{a}}$ with the original distribution will not only correct the non-uniform Fourier coefficient at $\boldsymbol{a}$ but also will not increase the non-uniform Fourier coefficients at any vector $\boldsymbol{b}$ as long as $\text{supp}(\boldsymbol{b}) \nsubseteq \text{supp}(\boldsymbol{a})$. Therefore we can start from vectors of weight k and correct the non-uniform Fourier coefficients level by level until we finish correcting vectors of weight 1 and finally obtain a k-wise independent distribution. Bounding the total weights added during this process gives an upper bound on the distance between D and non-uniform k-wise independence. The notion of non-uniform Fourier coefficients may find other applications when non-uniform independence is involved.

1.3 Other Related Research

There are many works on k-wise independence, most focus on various *constructions* of k-wise independence or distributions that approximate k-wise independence. k-wise independent random variables were first studied in probability theory [23] and then in complexity theory [13,2,28,29] mainly for derandomization purposes. Constructions of almost k-wise independent distributions were studied in [31,3,6,17,10]. Construction results of non-uniform k-wise independent distributions were given in [24,26].

There has been much activity on property testing of distributions. Some examples include testing uniformity [20,8], independence [7], monotonicity and being unimodal [9], estimating the support sizes [34] and testing a weaker notion than k-wise independence, namely, "almost k-wise independence" [1].

Many other techniques have also been developed to generalize results from Boolean domains to arbitrary domains [15,30,11].

1.4 Organization

We first give some necessary definitions in Section 2. Then we study k-wise independent distributions over general domains and product spaces in Section 3.1 and Section 3.2, respectively. The cases of non-uniform k-wise independence are treated in Section 4. Most proofs are omitted from this extended abstract, which may be found in the full version of the present paper.

2 Preliminaries

Let n and m be two natural numbers with $m > n$. We write $[n]$ for the set $\{1, \ldots, n\}$ and $[n, m]$ for the set $\{n, n+1, \ldots, m\}$. Throughout this paper, Σ always stands for a finite set. Without loss of generality, we assume that $\Sigma = \{0, 1, \ldots, q-1\}$, where $q = |\Sigma|$.

We use $\boldsymbol{a}$ to denote a vector $(a_1, \ldots, a_n)$ in Σ^n with a_i being the i^{th} component of $\boldsymbol{a}$. The support of $\boldsymbol{a}$ is the set of indices at which $\boldsymbol{a}$ is non-zero. That is, $\mathrm{supp}(\boldsymbol{a}) = \{i \in [n] : a_i \neq 0\}$. The weight of a vector $\boldsymbol{a}$ is the cardinality of its support. Let $1 \leq k \leq n$ be an integer. We use $M(n, k, q) := \binom{n}{1}(q-1) + \cdots + \binom{n}{k}(q-1)^k$ to denote the total number of non-zero vectors in Σ^n of weight at most k. Note that $M(n, k, q) = \Theta(n^k(q-1)^k)$ for $k = O(1)$. For two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ in Σ^n, we define their inner-product to be $\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i \pmod{q}$.

Let D_1 and D_2 be two distributions over the same domain $\mathfrak{D}$. The statistical distance between D_1 and D_2 is $\Delta(D_1, D_2) = \frac{1}{2}\sum_{x \in \mathfrak{D}} |D_1(x) - D_2(x)|$. One can check that statistical distance is a metric and in particular satisfies the triangle inequality. We use statistical distance as the main metric to measure closeness between distributions in this paper. For any $0 \leq \epsilon \leq 1$, define a new distribution to be the convex combination of D_1 and D_2 as $D' = \frac{1}{1+\epsilon}D_1 + \frac{\epsilon}{1+\epsilon}D_2$, then $\Delta(D', D_1) \leq \frac{\epsilon}{1+\epsilon} \leq \epsilon$. Sometimes we abuse notation and call the non-negative function ϵD_1 a *weighted-ϵ* distribution (in particular a *small-weight distribution* if ϵ is small).

Let $S = \{i_1, \ldots, i_k\} \subseteq [n]$ be an index set. The *projection distribution* of D with respect to S, denoted by D_S, is the distribution obtained by restricting to the coordinates in S. Namely, $D_S : \Sigma^k \to [0, 1]$ such that $D_S(z_{i_1} \cdots z_{i_k}) = \sum_{\boldsymbol{x} \in \Sigma^n : x_{i_1} = z_{i_1}, \ldots, x_{i_k} = z_{i_k}} D(\boldsymbol{x})$. For brevity, we sometimes write $D_S(z_j : j \in S)$ for $D_S(z_{i_1} \cdots z_{i_k})$. We also use $\boldsymbol{x}_S$ to denote the k-dimensional vector obtained from projecting $\boldsymbol{x}$ to the indices in S.

The k-wise Independent Distributions: Let $D : \Sigma_1 \times \cdots \times \Sigma_n \to [0, 1]$ be a distribution. We say D is a *uniform* distribution if for every $\boldsymbol{x} \in \Sigma_1 \times \cdots \times \Sigma_n$, $\Pr_{X \sim D}[X = \boldsymbol{x}] = \frac{1}{q_1 \cdots q_n}$, where $q_i = |\Sigma_i|$. D is *k-wise independent* if for any set of k indices $\{i_1, \ldots, i_k\}$ and for any $z_1 \cdots z_k \in \Sigma_{i_1} \times \cdots \times \Sigma_{i_k}$, $\Pr_{X \sim D}[X_{i_1} \cdots X_{i_k} = z_1 \cdots z_k] = \Pr_{X \sim D}[X_{i_1} = z_1] \times \cdots \times \Pr_{X \sim D}[X_{i_k} = z_k]$. D is *uniform k-wise independent* if, on top of the previous condition, we have $\Pr_{X \sim D}[X_i = z_j] = \frac{1}{|\Sigma_i|}$ for every i and every $z_j \in \Sigma_i$. Let D_{kwi} denote the set of all uniform k-wise independent distributions. The distance between D and D_{kwi}, denoted by $\Delta(D, D_{\text{kwi}})$, is the minimum statistical distance between D and any uniform k-wise independent distribution, i.e., $\Delta(D, D_{\text{kwi}}) := \min_{D' \in D_{\text{kwi}}} \Delta(D, D')$.

Discrete Fourier Transforms: For background on discrete Fourier transforms in computer science, the reader is referred to [39,40]. Let $f : \Sigma_1 \times \cdots \times \Sigma_n \to \mathbb{C}$ be any function defined over the discrete product space, we define the Fourier transform of D as, for all $\boldsymbol{a} \in \Sigma_1 \times \cdots \times \Sigma_n$,

$$\hat{f}(\boldsymbol{a}) = \sum_{\boldsymbol{x} \in \Sigma_1 \times \cdots \times \Sigma_n} f(\boldsymbol{x}) e^{2\pi i (\frac{a_1 x_1}{q_1} + \cdots + \frac{a_n x_n}{q_n})}.$$

One can easily verify that the inverse Fourier transform is

$$f(\boldsymbol{x}) = \frac{1}{q_1 \cdots q_n} \sum_{\boldsymbol{a} \in \Sigma_1 \times \cdots \times \Sigma_n} \hat{f}(\boldsymbol{a}) e^{-2\pi i (\frac{a_1 x_1}{q_1} + \cdots + \frac{a_n x_n}{q_n})}.$$

Note that if $\Sigma_i = \Sigma$ for every $1 \leq i \leq n$ (which is the main focus of this paper), then $\hat{f}(\boldsymbol{a}) = \sum_{\boldsymbol{x} \in \Sigma^n} f(\boldsymbol{x}) e^{\frac{2\pi i}{q} \boldsymbol{a} \cdot \boldsymbol{x}}$ and $f(\boldsymbol{x}) = \frac{1}{|\Sigma|^n} \sum_{\boldsymbol{a} \in \Sigma^n} \hat{f}(\boldsymbol{a}) e^{-\frac{2\pi i}{q} \boldsymbol{a} \cdot \boldsymbol{x}}$.

We will use the following two simple facts about Fourier transforms which are easy to verify.

Fact 1. *For any integer ℓ which is not congruent to 0 modulo q, $\sum_{j=0}^{q-1} e^{\frac{2\pi i}{q} \ell j} = 0$.*

Fact 2. *Let d, ℓ_0 be integers such that $d|q$ and $0 \leq \ell_0 \leq d-1$. Then $\sum_{\ell=0}^{\frac{q}{d}-1} e^{\frac{2\pi i}{q}(\ell_0 + d\ell)} = 0$.*

Proposition 1. *Let D be a distribution over $\Sigma_1 \times \cdots \times \Sigma_n$. Then D is a uniform distribution if and only if for any non-zero vector $\boldsymbol{a} \in \Sigma_1 \times \cdots \times \Sigma_n$, $\hat{D}(\boldsymbol{a}) = 0$.*

By applying Proposition 1 to distributions obtained from restriction to any k indices, we have the following characterization of k-wise independent distributions over product spaces, which is the basis of all of our testing algorithms.

Corollary 1. *A distribution D over $\Sigma_1 \times \cdots \times \Sigma_n$ is k-wise independent if and only if for all non-zero vectors $\boldsymbol{a}$ of weight at most k, $\hat{D}(\boldsymbol{a}) = 0$.*

Other Definitions and Notation: We are going to use the following notation extensively in this paper.

Definition 1. *Let D be a distribution over Σ^n. For every $\boldsymbol{a} \in \Sigma^n$ and every $0 \leq j \leq q-1$, define $P^D_{\boldsymbol{a},j} := \Pr_{X \sim D}[\boldsymbol{a} \cdot \boldsymbol{X} \equiv j \pmod{\text{q}}]$. When the distribution D is clear from context, we often omit the superscript D and simply write $P_{\boldsymbol{a},j}$.*

For any non-zero vector $\boldsymbol{a} \in \mathbb{Z}_q^n$ and any integer j, $0 \leq j \leq q-1$, let $S_{\boldsymbol{a},j} := \{X \in \mathbb{Z}_q^n : \sum_{i=1}^n a_i X_i \equiv j \pmod{\text{q}}\}$. Let $U_{\boldsymbol{a},j}$ denote the uniform distribution over $S_{\boldsymbol{a},j}$.

3 Uniform k-Wise Independent Distributions over Product Spaces

3.1 Domains of the Form $\mathbb{Z}_q^n$

We first consider the problem of testing k-wise independent distributions over domains of the form $\mathbb{Z}_q^n$, where q is the size of the alphabet. Recall that a distribution D over $\mathbb{Z}_q^n$ is k-wise independent if and only if for all non-zero

vectors $\boldsymbol{a}$ of weight at most k, $\hat{D}(\boldsymbol{a}) = 0$. In the following, we are going to show that we can mix D with a series (small-weight) distributions to get a new distribution D' such that $\hat{D}'(\boldsymbol{a}) = 0$ for every $0 < \mathrm{wt}(\boldsymbol{a}) \leq k$. Therefore D' is k-wise independent and thus the total weights of the distributions used for mixing is an upper bound on the distance between D and the set of k-wise independent distributions.

Unless stated otherwise, all arithmetic operations in this section are performed modulo q; For instance, we use $\boldsymbol{a} = \boldsymbol{b}$ to mean that $a_i \equiv b_i \pmod{\mathrm{q}}$ for each $1 \leq i \leq n$.

Let $\boldsymbol{a} = (a_1, \ldots, a_n)$ be a non-zero vector. We say $\boldsymbol{a}$ is a *prime vector* if $\gcd(a_1, \ldots, a_n) = 1$. If $\boldsymbol{a}$ is a prime vector, then we refer to the set of vectors $\{2\boldsymbol{a}, \ldots, (q-1)\boldsymbol{a}\}$ (note that all these vectors are distinct) as the *siblings* of $\boldsymbol{a}$, and together with $\boldsymbol{a}$ collectively we refer to them as a *family* of vectors. Note that families of vectors do *not* form a partition of all the vectors. For example when $n = 2$ and $q = 6$, vector $(4, 0)$ is a sibling of both $(1, 0)$ and $(2, 3)$, but the latter two vectors are not siblings of each other.

Recall that $S_{\boldsymbol{a},j}$ denotes the set $\{x \in \mathbb{Z}_q^n : \sum_{i=1}^n a_i x_i \equiv j \pmod{\mathrm{q}}\}$.

Proposition 2. *If $\boldsymbol{a}$ is a prime vector, then $|S_{\boldsymbol{a},j}| = q^{n-1}$ for any $0 \leq j \leq q-1$.*

Linear Systems of Congruences. Here we record some useful results on linear systems of congruences. For more on this, the interested reader is referred to [22] and [38]. These results will be used in the next section to show some important orthogonality properties of vectors. In this section, all matrices are integer-valued. Let M be a $k \times n$ matrix with $k \leq n$. The *greatest divisor* of M is the greatest common divisor (gcd) of the determinants of all the $k \times k$ sub-matrices of M. M is a *prime matrix* if the greatest divisor of M is 1.

Lemma 1 ([38]). *Let M be a $(k+1) \times n$ matrix. If the sub-matrix consisting of the first k rows of M is a prime matrix and M has greatest divisor d, then there exist integers $u_1, \ldots, u_k$ such that*

$$u_1 M_{1,j} + u_2 M_{2,j} + \ldots + u_k M_{k,j} \equiv M_{k+1,j} \pmod{\mathrm{d}},$$

for every $1 \leq j \leq n$.

Consider the following system of linear congruent equations:

$$\begin{cases} M_{1,1}x_1 + M_{1,2}x_2 + \cdots + M_{1,n}x_n \equiv M_{1,n+1} \pmod{\mathrm{q}} \\ \qquad\qquad\qquad\qquad \vdots \\ M_{k,1}x_1 + M_{k,2}x_2 + \cdots + M_{k,n}x_n \equiv M_{k,n+1} \pmod{\mathrm{q}}. \end{cases} \tag{1}$$

Let M denote the $k \times n$ matrix consisting of the coefficients of the linear system of equations and let $\tilde{M}$ denote the corresponding augmented matrix of M, that is, the $k \times (n+1)$ matrix including the extra column of constants.

Definition 2. *Let M be the coefficient matrix of Eq. (1) and $\tilde{M}$ be the augmented matrix. Suppose $k < n$ so that system (1) is a defective system of equations. Define $Y_k, Y_{k-1}, \ldots, Y_1$ respectively to be the greatest common divisors of the determinants of all the $k \times k, (k-1) \times (k-1), \ldots, 1 \times 1$, respectively sub-matrices of M.*

Similarly define $Z_k, Z_{k-1}, \ldots, Z_1$ *for the augmented matrix* $\tilde{M}$. *Also we define* $Y_0 = 1$ *and* $Z_0 = 1$. *Define* $s = \prod_{j=1}^{k} \gcd(q, \frac{Y_j}{Y_{j-1}})$ *and* $t = \prod_{j=1}^{k} \gcd(q, \frac{Z_j}{Z_{j-1}})$.

The following theorem of Smith gives the necessary and sufficient conditions for a system of congruent equations to have solutions.

Theorem 3 ([38]). *If* $k < n$, *then the (defective) linear system of congruences (1) has solutions if and only if* $s = t$. *Moreover, if this condition is met, the number of incongruent solutions is* sq^{n-k}.

Weak Orthogonality between Families of Vectors. To generalize the proof idea of the GF(2) case to commutative rings $\mathbb{Z}_q$ for arbitrary q, it seems crucial to relax the requirement that linearly independent vectors are strongly orthogonal. Rather, we introduce the notion of weak orthogonality between a pair of vectors.

Definition 3. *Let* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *be two vectors in* $\mathbb{Z}_q^n$. *We say* $\boldsymbol{a}$ *is* weakly orthogonal *to* $\boldsymbol{b}$ *if for all* $0 \leq j \leq q-1$, $\hat{U}_{\boldsymbol{a},j}(\boldsymbol{b}) = 0$.

We remark that strong orthogonality (defined in the Introduction) implies weak orthogonality while the converse is not necessarily true. In particular, strong orthogonality does not hold in general for linearly independent vectors in $\mathbb{Z}_q^n$. However, for our purpose of constructing k-wise independent distributions, weak orthogonality between pairs of vectors suffices.

The following Lemma is the basis of our upper bound on the distance from a distribution to k-wise independence. This Lemma enables us to construct a small-weight distribution using an appropriate convex combination of $\{U_{\boldsymbol{a},j}\}_{j=0}^{q-1}$, which on the one hand zeros-out all the Fourier coefficients at $\boldsymbol{a}$ and its sibling vectors, on the other hand has zero Fourier coefficient at all other vectors. The proof of the Lemma relies crucially on the results in Section 3.1 about linear system of congruences.

Lemma 2 (Main). *Let* $\boldsymbol{a}$ *be a non-zero prime vector and* $\boldsymbol{b}$ *any non-zero vector that is not a sibling of* $\boldsymbol{a}$. *Then* $\boldsymbol{a}$ *is weakly orthogonal to* $\boldsymbol{b}$.

Proof. Consider the following system of linear congruences:

$$\begin{cases} a_1x_1 + a_2x_2 + \cdots + a_nx_n \equiv a_0 \pmod{q} \\ b_1x_1 + b_2x_2 + \cdots + b_nx_n \equiv b_0 \pmod{q}. \end{cases} \tag{2}$$

Following our previous notation, let $M = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ b_1 & b_2 & \cdots & b_n \end{bmatrix}$ and $\tilde{M} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n & a_0 \\ b_1 & b_2 & \cdots & b_n & b_0 \end{bmatrix}$. Since $\boldsymbol{a}$ is a prime vector, $Y_1 = Z_1 = 1$. We next show that Y_2 can not be a multiple of q.

Claim. Let $M = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ b_1 & b_2 & \cdots & b_n \end{bmatrix}$. The determinants of all 2×2 sub-matrices of M are congruent to 0 modulo q if and only if $\boldsymbol{a}$ and $\boldsymbol{b}$ are sibling vectors.

Proof. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are sibling vectors, then it is clear that the determinants of all the sub-matrices are congruent to 0 modulo q. For the *only if* direction, we may assume that $\boldsymbol{a}$ is a prime vector, since otherwise we can divide the first row of M by the common divisor. all we need to prove is that $\boldsymbol{b} = c\boldsymbol{a}$ for some integer c. First suppose that the determinants of all 2×2 sub-matrices of M are 0. Then it follows that $\frac{b_1}{a_1} = \cdots = \frac{b_n}{a_n} = c$. If c is not an integer, then $c = \frac{u}{v}$, where u, v are integers and $\gcd(u, v) = 1$. But this implies $v|a_i$ for every $1 \leq i \leq n$, contradicting our assumption that $\boldsymbol{a}$ is a prime vector. Now if not all of the determinants are 0, it must be the case that the greatest divisor of the determinants of all 2×2 sub-matrices, say d', is a multiple of q. By Lemma 1, there is an integer c such that $ca_i \equiv b_i \pmod{\mathrm{d}'}$ for every $1 \leq i \leq n$. Consequently, $b_i \equiv ca_i \pmod{\mathrm{q}}$ for every i and hence $\boldsymbol{b}$ is a sibling of $\boldsymbol{a}$. □

Let $d = \gcd(q, Y_2)$. Clearly $1 \leq d \leq q-1$ and, according to Claim 3.1, $d|q$. Applying Theorem 3 with $k = 2$ to (2), the two linear congruences are solvable if and only if $d = \gcd(q, Y_2) = \gcd(q, Z_2)$. If this is the case, the total number of incongruent solutions is dq^{n-2}. Furthermore, if we let h denote the greatest common divisor of the determinants of all 2×2 sub-matrices of $\tilde{M}$, then $d|h$. By Lemma 1, there is an integer u such that $b_0 \equiv ua_0 \pmod{\mathrm{h}}$. It follows that $d|(b_0 - ua_0)$. Let us consider a fixed a_0 and write $\ell_0 = ua_0 \pmod{d}$. Since $\boldsymbol{a}$ is a prime vector, by Proposition 2, there are in total q^{n-1} solutions to (2). But for any specific b_0 that has solutions to (2), there must be dq^{n-2} solutions to (2) and in addition $d|q$. Since there are exactly q/d b_0's in $\{0, \ldots, q-1\}$, we conclude that (2) has solutions for b_0 if and only if $b_0 = \ell_0 + d\ell$, where ℓ_0 is some constant and $\ell = 0, \ldots, \frac{q}{d} - 1$. Finally we have

$$\begin{aligned}\hat{U}_{\boldsymbol{a},j}(\boldsymbol{b}) &= \sum_{\boldsymbol{x} \in \mathbb{Z}_q^n} U_{\boldsymbol{a},j}(\boldsymbol{x}) e^{\frac{2\pi i}{q} \boldsymbol{b}\cdot\boldsymbol{x}} = \frac{1}{q^{n-1}} \sum_{\boldsymbol{a}\cdot\boldsymbol{x} \equiv j \pmod{\mathrm{q}}} e^{\frac{2\pi i}{q} \boldsymbol{b}\cdot\boldsymbol{x}} \\ &= \frac{d}{q} \sum_{b_0 : b_0 = \ell_0 + d\ell} e^{\frac{2\pi i}{q} b_0} = 0. \qquad \text{(by Fact 2)}\end{aligned}$$

This finishes the proof of Lemma 2. □

Correcting Fourier Coefficients of Sibling Vectors. In this section, we show how to zero-out all the Fourier coefficients of a family of vectors. Let D be a distribution over $\mathbb{Z}_q^n$. Note that, for every $1 \leq \ell \leq q-1$, the Fourier coefficient of a vector $\ell\boldsymbol{a}$ can be rewritten as $\hat{D}(\ell\boldsymbol{a}) = \sum_{\boldsymbol{x} \in G} D(\boldsymbol{x}) e^{\frac{2\pi i}{q} \ell \boldsymbol{a}\cdot\boldsymbol{x}} = \sum_{j=0}^{q-1} \Pr_{x \sim D}[\boldsymbol{a} \cdot \boldsymbol{x} \equiv j \pmod{\mathrm{q}}] e^{\frac{2\pi i}{q} \ell j} = \sum_{j=0}^{q-1} P_{\boldsymbol{a},j} e^{\frac{2\pi i}{q} \ell j}$. Define $\mathrm{MaxBias}(\boldsymbol{a}) := \max_{0 \leq j \leq q-1} P_{\boldsymbol{a},j} - \frac{1}{q}$.

Claim. We have that $\mathrm{MaxBias}(\boldsymbol{a}) \leq \frac{1}{q} \sum_{\ell=1}^{q-1} |\hat{D}(\ell\boldsymbol{a})|$.

Theorem 4. *Let D be a distribution over $\mathbb{Z}_q^n$, then*[4]

$$\Delta(D, D_{\text{kwi}}) \leq \sum_{0<\text{wt}(\boldsymbol{a})\leq k} |\hat{D}(\boldsymbol{a})|.$$

In particular, $\Delta(D, D_{\text{kwi}}) \leq M(n,k,q) \max_{0<\text{wt}(\boldsymbol{a})\leq k} |\hat{D}(\boldsymbol{a})|$.

Testing Algorithm and its Analysis. The following Theorem summarizes the query and time complexities of our testing algorithm for uniform k-wise independence. The proof is omitted here due to space limitation.

Theorem 5. *There is an algorithm that tests k-wise independence over $\{0,\ldots,q-1\}^n$ with query complexity $\tilde{O}(\frac{n^{2k}(q-1)^{2k}q^2}{\epsilon^2})$ and time complexity $\tilde{O}(\frac{n^{3k}(q-1)^{3k}q^2}{\epsilon^2})$.*

3.2 Uniform k-Wise Independent Distributions over Product Spaces

Now we generalize the $\mathbb{Z}_q^n$ domains case to product spaces. Let $\Sigma_1,\ldots,\Sigma_n$ be finite sets. Without loss of generality, let $\Sigma_i = \{0,1,\ldots,q_i-1\}$. In this section, we consider distributions over product space $\Sigma_1 \times \cdots \times \Sigma_n$. Let $M = \text{lcm}(q_1,\ldots,q_n)$ and for each $1 \leq i \leq n$ let $M_i = \frac{M}{q_i}$. Then the Fourier coefficients can be rewritten as $\hat{D}(\boldsymbol{a}) = \sum_{\boldsymbol{x}\in\Sigma_1\times\cdots\times\Sigma_n} D(\boldsymbol{x}) e^{\frac{2\pi i}{M}(M_1 a_1 x_1 + \cdots + M_n a_n x_n)} = \sum_{\boldsymbol{x}\in\Sigma_1\times\cdots\times\Sigma_n} D(\boldsymbol{x}) e^{\frac{2\pi i}{M}(a'_1 x_1 + \cdots + a'_n x_n)}$, where $a'_i = M_i a_i$. Therefore we can see D as a distribution over Σ^n with *effective alphabet size* $|\Sigma| = M = \text{lcm}(q_1,\ldots,q_n)$ and we are only concerned with Fourier coefficients at $\boldsymbol{a'} = (a'_1,\ldots,a'_n)$. Note that in general $M = \text{lcm}(q_1,\ldots,q_n)$ can be an exponentially large number and is therefore not easy to handle in practice[5]. We overcome this difficulty by observing that, since we are only concerned with vectors of weight at most k, we may take different effective alphabet sizes for different index subsets of size k, i.e., $|\Sigma_S| = \text{lcm}(q_{i_1},\ldots,q_{i_k})$ where $S = \{i_1,\ldots,i_k\}$.

Under this formalism, we can prove the following Theorem:

Theorem 6. *Let D be a distribution over $\Sigma_1 \times \cdots \times \Sigma_n$. Then* $\Delta(D, D_{\text{kwi}}) \leq \sum_{0<\text{wt}(\boldsymbol{a})\leq k} |\hat{D}(\boldsymbol{a})|$.

4 Non-uniform k-Wise Independent Distributions

In this section we focus on non-uniform k-wise independent distributions. For ease of exposition, we only prove our results for the case when the underlying

[4] It is easy to verify that the same bound holds for prime field case if we transform the bound in MaxBias there into a bound in terms of Fourier coefficients. Conversely we can equivalently write the bound of the distance from k-wise independence in terms of MaxBias over *prime vectors*. However, we believe that stating the bound in terms of Fourier coefficients is more natural and generalizes more easily.

[5] Recall that the testing algorithm requires estimating all the low-degree Fourier coefficients which is an exponential sum with M as the denominator.

domain is Σ^n with $q = |\Sigma|$. Our approach here generalizes easily to distributions over product spaces.

Recall that a distribution $D : \Sigma^n \to [0,1]$ is k-wise independent if for any index subset $S \subset [n]$ of size k, $S = \{i_1, \ldots, i_k\}$, and for any $z_1 \cdots z_k \in \Sigma^k$, $D_S(z_1 \cdots z_k) = \Pr_D[X_{i_1} = z_1] \cdots \Pr_D[X_{i_k} = z_k]$. We prove an upper bound on the distance between D and k-wise independence by reducing the problem to uniform case and then applying Theorem 4.

In the following we define a set of multipliers which are used to transform non-uniform k-wise independent distributions into uniform ones. Let $p_i(z) := \Pr_D[X_i = z]$. We assume that $0 < p_i(z) < 1$ for all $i \in [n]$ and $z \in \Sigma$ (this is without loss of generality since if some $p_i(z)$'s are zero, then it reduces to the case of distributions over product spaces). Define $\theta_i(z) := \frac{1}{qp_i(z)}$. Intuitively, one may think $\theta_i(z)$ as a set of compressing/stretching factors which transform a non-uniform k-wise distribution into a uniform one. For notation convenience, if $S = \{i_1, \ldots, i_\ell\}$ and $\boldsymbol{z} = z_{i_1} \cdots z_{i_\ell}$, we use $\theta_S(\boldsymbol{z})$ to denote the product $\theta_{i_1}(z_{i_1}) \cdots \theta_{i_\ell}(z_{i_\ell})$.

Definition 4 (Non-uniform Fourier Coefficients). *Let D be a distribution over Σ^n. Let $\boldsymbol{a}$ be a non-zero vector in Σ^n with* $\mathrm{supp}(\boldsymbol{a})$ *being its support set and $D_{\mathrm{supp}(\boldsymbol{a})}$ be the projection distribution of D with respect to* $\mathrm{supp}(\boldsymbol{a})$*. Set $D'_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{z}) = \theta_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{z}) D_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{z})$, which is the transformed distribution*[6] *of the projection distribution $D_{\mathrm{supp}(\boldsymbol{a})}$. Then the* non-uniform Fourier coefficient *of D at $\boldsymbol{a}$ is*

$$\hat{D}^{non}(\boldsymbol{a}) = \hat{D'}_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{a}) = \sum_{\boldsymbol{z} \in \Sigma^{\mathrm{supp}(\boldsymbol{a})}} D'_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{z}) e^{\frac{2\pi i}{q} \boldsymbol{a} \cdot \boldsymbol{z}}. \tag{3}$$

The idea of defining $D'_{\mathrm{supp}(\boldsymbol{a})}$ is that, if D is non-uniform k-wise independent, then $D'_{\mathrm{supp}(\boldsymbol{a})}$ will be a uniform distribution over the index set $\mathrm{supp}(\boldsymbol{a})$. Indeed, our main result in this section is to show the connection between non-uniform Fourier coefficients and the property of the distribution D being k-wise independent. In particular we have the following simple characterization of non-uniform k-wise independence.

Theorem 7. *A distribution D over Σ^n is k-wise independent if and only if for every non-zero vector $\boldsymbol{a} \in \Sigma^k$ with $\mathrm{wt}(\boldsymbol{a}) \leq k$, $\hat{D}^{non}(\boldsymbol{a}) = 0$.*

The proof of Theorem 7 relies on the observation that, when written in the form of linear transformation, non-uniform Fourier transform matrix, like the uniform Fourier transform matrix, can be expressed as a tensor product of a set of heterogeneous DFT (discrete Fourier transform) matrices. This enables us to show that the non-uniform Fourier transform is invertible.

Given a distribution D which is not k-wise independent, what is its distance to non-uniform k-wise independence? In the following, we will follow the same

[6] Note that in general $D'_{\mathrm{supp}(\boldsymbol{a})}$ is not a distribution, since although it is non-negative everywhere but $\sum_{\boldsymbol{x}} D'_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{x}) = 1$ may not hold.

approach as used in the uniform case and try to find a set of small-weight distributions to mix with D to zero-out all the non-uniform Fourier coefficients at vectors of weight at most k. This will show the robustness of characterization of non-uniform k-wise independence given in Theorem 7.

A careful inspection of Theorem 4 and its proof shows that, if we focus on the weights added to correct any fixed prime vector and its siblings, we actually prove the following.

Theorem 8. *Let E' be a distribution over Σ^n, $\boldsymbol{a}$ be a prime vector of weight at most k and let $\hat{E}'(\boldsymbol{a}), \ldots, \hat{E}'((q-1)\boldsymbol{a})$ be the Fourier coefficients at $\boldsymbol{a}$ and its sibling vectors. Then there exist a set of non-negative real numbers $w_j, j = 0, 1, \ldots, q-1$ such that the (small-weight) distribution[7] $U_{E',\boldsymbol{a}} = \sum_{j=0}^{q-1} w_j U_{\boldsymbol{a},j}$ has the following properties. $\hat{U}_{E',\boldsymbol{a}}(\boldsymbol{b}) = 0$ for all non-zero vectors that are not siblings of $\boldsymbol{a}$ and $E' + U_{E',\boldsymbol{a}}$ has zero Fourier coefficients at $\boldsymbol{a}, 2\boldsymbol{a}, \ldots, (q-1)\boldsymbol{a}$. Moreover, $\sum_{j=0}^{q-1} w_j \leq \sum_{\ell=1}^{q-1} |\hat{E}'(\ell\boldsymbol{a})|$.*

It is easy to see that the Theorem applies to any non-negative functions as well. Applying Theorem 8 with E' equal to $D'_{\mathrm{supp}(\boldsymbol{a})}$ gives rise to a small-weight distribution $U_{\mathrm{supp}(\boldsymbol{a}),\boldsymbol{a}}$ which we denote by $U_{\boldsymbol{a}}$, to zero-out all the Fourier coefficients at $\boldsymbol{a}$ and its siblings[8]. Now we apply the (reversed) compressing/stretching factor to $U_{\boldsymbol{a}}$ to get $\tilde{U}_{\boldsymbol{a}}$,

$$\tilde{U}_{\boldsymbol{a}}(\boldsymbol{x}) = \frac{U_{\boldsymbol{a}}(\boldsymbol{x})}{\theta_{[n]}(\boldsymbol{x})}. \tag{4}$$

The following Lemma shows that mixing with $\tilde{U}_{\boldsymbol{a}}$ zeroes-out the D's non-uniform Fourier coefficients at $\boldsymbol{a}$ and its sibling vectors. Moreover, the mixing only adds up a relative small amount of weight and can only mess up the non-uniform Fourier coefficient at vectors whose support sets are strictly contained in the support set of $\boldsymbol{a}$.

Lemma 3. *Let D be a distribution over Σ^n and $\boldsymbol{a}$ be a prime vector of weight at most k. Let $\mathrm{supp}(\boldsymbol{a})$ be the support set of $\boldsymbol{a}$ and $\tilde{U}_{\boldsymbol{a}}$ be as defined in Equation (4). Let $\gamma_k := \max_{S,\boldsymbol{z}} \frac{1}{\theta_S(\boldsymbol{z})}$, where S is a subset of $[n]$ of size at most k and $\boldsymbol{z} \in \Sigma^{|S|}$. Then*

- *The non-uniform Fourier coefficients of $D + \tilde{U}_{\boldsymbol{a}}$ at $\boldsymbol{a}$ as well as at the sibling vectors of $\boldsymbol{a}$ whose support sets are also $\mathrm{supp}(\boldsymbol{a})$ are all zero. Moreover, $\hat{\tilde{U}}^{\mathrm{non}}_{\boldsymbol{a}}(\boldsymbol{a}') = 0$ for every vector $\boldsymbol{a}'$ whose support set is $\mathrm{supp}(\boldsymbol{a})$ but is not a sibling vector of $\boldsymbol{a}$.*
- *For any vector $\boldsymbol{b}$ with $\mathrm{supp}(\boldsymbol{b}) \not\subseteq \mathrm{supp}(\boldsymbol{a})$, $\hat{\tilde{U}}^{\mathrm{non}}_{\boldsymbol{a}}(\boldsymbol{b}) = 0$.*
- *The total weight of $\tilde{U}_{\boldsymbol{a}}$ is at most $\gamma_k \sum_{\boldsymbol{x} \in \Sigma^n} U_{\boldsymbol{a}}(\boldsymbol{x}) \leq \gamma_k \sum_{j=1}^{q-1} |\hat{D}^{\mathrm{non}}(j\boldsymbol{a})|$.*

[7] Recall that $U_{\boldsymbol{a},j}$ is the uniform distribution over all strings $x \in \mathbb{Z}_q^n$ with $\boldsymbol{a} \cdot \boldsymbol{x} \equiv j \pmod{\mathrm{q}}$.

[8] In fact, this only guarantees to zero-out the Fourier coefficients at $\boldsymbol{a}$ and its siblings whose support sets are the same as that of $\boldsymbol{a}$. But that suffices for our correcting purposes because we will proceed to vectors with smaller support sets in later stages.

Algorithm `Testing Non-uniform k-wise Independence` (D, k, q, ϵ)

1. Sample D uniformly and independently M times
2. Use the samples to estimate, for each non-zero vector $\boldsymbol{a}$ of weight at most k and each $\boldsymbol{z}$, $D_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{z})$, where $\mathrm{supp}(\boldsymbol{a})$ is the support set of $\boldsymbol{a}$
   - Compute $D'_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{z}) = \theta_S(\boldsymbol{z}) D_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{z})$
   - Compute $\hat{D}^{\mathrm{non}}(\boldsymbol{a}) = \hat{D'}_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{a}) = \sum_{\boldsymbol{z}} D'_{\mathrm{supp}(\boldsymbol{a})}(\boldsymbol{z}) e^{\frac{2\pi i}{q} \boldsymbol{a} \cdot \boldsymbol{z}}$ for each $\boldsymbol{a} \in \Sigma^k$
3. If $\max_{\boldsymbol{a}} |\hat{D}^{\mathrm{non}}(\boldsymbol{a})| \leq \delta$ return **"Yes"**; else return **"No"**

Fig. 2. Algorithm for testing non-uniform k-wise independence

- *For any non-zero vector* $\boldsymbol{c}$ *with* $\mathrm{supp}(\boldsymbol{c}) \subset \mathrm{supp}(\boldsymbol{a})$, $\hat{\hat{U}}^{\mathrm{non}}_{\boldsymbol{a}}(\boldsymbol{c}) \leq \gamma_k \sum_{j=1}^{q-1} |\hat{D}^{\mathrm{non}}(j\boldsymbol{a})|$.

Now we can, for each prime vector $\boldsymbol{a}$ whose support set is of size k, mix D with $\tilde{U}_{\boldsymbol{a}}$ to zero-out all the level k non-uniform Fourier coefficients. By Lemma 3 these added weights can only mess up the non-uniform Fourier coefficients at level less than k. We then recompute the non-uniform Fouriere coefficients of the new distribution and repeat this process for vectors whose support sets are of size $k-1$. Keep doing this until zeroing-out all the non-uniform Fourier coefficients at vectors of weight 1, we finally obtain a non-uniform k-wise independent distribution.

Theorem 9. *Let D be a distribution over Σ^n, then*

$$\Delta(D, D_{\mathrm{kwi}}) \leq O\left(n^k q^{\frac{k(k+3)}{2}}\right) \max_{\boldsymbol{a}: 0 < \mathrm{wt}(\boldsymbol{a}) \leq k} |\hat{D}^{non}(\boldsymbol{a})|.$$

4.1 Testing Algorithm and Its Analysis

In Fig. 2, we give an outline of the algorithm for testing non-uniform k-wise independence when all the marginal probabilities $p_i(z)$ are assumed to be known.[9] The analysis of the testing algorithm is very much the same[10] as that presented in Section 3.1, we leave the details to interested readers.

Acknowledgments

We would like to thank Per Austrin for correspondence on constructing orthogonal real functions, Tali Kaufman for useful discussions and suggestions and Elchanan Mossel for his enthusiasm in this problem and a enlightening conversation. We are grateful to the anonymous referees for pointing out a critical error in an earlier version of this paper and for many helpful comments.

[9] If we assume that these probabilities are all bounded away from 0 or 1, then they can also be estimated from a small number of samples drawn independently from the distribution.

[10] One major difference is that we need to use the following simple fact to show completeness of the testing algorithm: if $\Delta(D, D_{\mathrm{kwi}}) \leq \delta$, then $|\hat{D}^{\mathrm{non}}(\boldsymbol{a})| \leq q\gamma_k\delta$ for all non-zero vectors $\boldsymbol{a}$ of weight at most k.

References

1. Alon, N., Andoni, A., Kaufman, T., Matulef, K., Rubinfeld, R., Xie, N.: Testing k-wise and almost k-wise independence. In: Proc. 39th Annual ACM Symposium on the Theory of Computing, pp. 496–505 (2007)
2. Alon, N., Babai, L., Itai, A.: A fast and simple randomized algorithm for the maximal independent set problem. Journal of Algorithms 7, 567–583 (1986)
3. Alon, N., Goldreich, O., Håstad, J., Peralta, R.: Simple constructions of almost k-wise independent random variables. Random Structures and Algorithms 3(3), 289–304 (1992); Earlier version in FOCS 1990
4. Alon, N., Goldreich, O., Mansour, Y.: Almost k-wise independence versus k-wise independence. Information Processing Letters 88, 107–110 (2003)
5. Austrin, P.: Conditional inapproximability and limited independence. PhD thesis, KTH - Royal Institute of Technology (2008), `http://www.csc.kth.se/~austrin/papers/thesis.pdf`
6. Azar, Y., Naor, J., Motwani, R.: Approximating probability distributions using small sample spaces. Combinatorica 18(2), 151–171 (1998)
7. Batu, T., Fischer, E., Fortnow, L., Kumar, R., Rubinfeld, R., White, P.: Testing random variables for independence and identity. In: Proc. 42nd Annual IEEE Symposium on Foundations of Computer Science, pp. 442–451 (2001)
8. Batu, T., Fortnow, L., Rubinfeld, R., Smith, W.D., White, P.: Testing that distributions are close. In: Proc. 41st Annual IEEE Symposium on Foundations of Computer Science, pp. 189–197 (2000)
9. Batu, T., Kumar, R., Rubinfeld, R.: Sublinear algorithms for testing monotone and unimodal distributions. In: Proc. 36th Annual ACM Symposium on the Theory of Computing, pp. 381–390. ACM Press, New York (2004)
10. Bertram-Kretzberg, C., Lefmann, H.: MOD_p-tests, almost independence and small probability spaces. Random Structures and Algorithms 16(4), 293–313 (2000)
11. Blais, E.: Testing juntas nearly optimally. In: Proc. 41st Annual ACM Symposium on the Theory of Computing, pp. 151–158 (2009)
12. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. Journal of Computer and System Sciences 47, 549–595 (1993); Earlier version in STOC 1900
13. Chor, B., Goldreich, O.: On the power of two-point based sampling. Journal of Complexity 5(1), 96–106 (1989)
14. Czumaj, A., Sohler, C.: Sublinear-time algorithms. Bulletin of the European Association for Theoretical Computer Science 89, 23–47 (2006)
15. Diakonikolas, I., Lee, H., Matulef, K., Onak, K., Rubinfeld, R., Servedio, R., Wan, A.: Testing for concise representations. In: Proc. 48th Annual IEEE Symposium on Foundations of Computer Science, pp. 549–558 (2007)
16. Efremenko, K.: 3-query locally decodable codes of subexponential length. In: Proc. 41st Annual ACM Symposium on the Theory of Computing, pp. 39–44 (2009)
17. Even, G., Goldreich, O., Luby, M., Nisan, N., Velickovic, B.: Efficient approximation of product distributions. Random Structures and Algorithms 13(1), 1–16 (1998); Earlier version in STOC 1992
18. Fischer, E.: The art of uninformed decisions: A primer to property testing. Bulletin of the European Association for Theoretical Computer Science 75 (2001)
19. Goldreich, O., Goldwaser, S., Ron, D.: Property testing and its connection to learning and approximation. Journal of the ACM 45, 653–750 (1998)

20. Goldreich, O., Ron, D.: On testing expansion in bounded-degree graphs. Technical Report TR00-020, Electronic Colloquium on Computational Complexity (2000)
21. Grolmusz, V.: Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs. Combinatorica 20(1), 71–86 (2000)
22. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers, 5th edn. Oxford University Press, Oxford (1980)
23. Joffe, A.: On a set of almost deterministic k-independent random variables. Annals of Probability 2, 161–162 (1974)
24. Karloff, H., Mansour, Y.: On construction of k-wise independent random variables. In: Proc. 26th Annual ACM Symposium on the Theory of Computing, pp. 564–573 (1994)
25. Karp, R., Wigderson, A.: A fast parallel algorithm for the maximal independent set problem. Journal of the ACM 32(4), 762–773 (1985)
26. Koller, D., Megiddo, N.: Constructing small sample spaces satisfying given constraints. In: Proc. 25th Annual ACM Symposium on the Theory of Computing, pp. 268–277 (1993)
27. Kumar, R., Rubinfeld, R.: Sublinear time algorithms. SIGACT News 34, 57–67 (2003)
28. Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM Journal on Computing 15(4), 1036–1053 (1986); Earlier version in STOC 1985
29. Luby, M.: Removing randomness in parallel computation without a processor penalty. In: Proc. 29th Annual IEEE Symposium on Foundations of Computer Science, pp. 162–173 (1988)
30. Mossel, E.: Gaussian bounds for noise correlation of functions and tight analysis of long codes. In: Proc. 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 156–165 (2008)
31. Naor, J., Naor, M.: Small-bias probability spaces: efficient constructions and applications. SIAM Journal on Computing 22(4), 838–856 (1993); Earlier version in STOC 1990
32. Neuman, C.P., Schonbach, D.I.: Discrete (Legendre) orthogonal polynomials - A survey. International Journal for Numerical Methods in Engineering 8, 743–770 (1974)
33. Nikiforov, A.F., Suslov, S.K., Uvarov, V.B.: Classical Orthogonal Polynomials of a Discrete Variable. Springer, Heidelberg (1991)
34. Raskhodnikova, S., Ron, D., Shpilka, A., Smith, A.: Strong lower bounds for approximating distribution support size and the distinct elements problem. In: Proc. 48th Annual IEEE Symposium on Foundations of Computer Science, pp. 559–569 (2007)
35. Ron, D.: Property testing (a tutorial). In: Pardalos, P.M., Rajasekaran, S., Reif, J., Rolim, J.D.P. (eds.) Handbook of Randomized Computing, pp. 597–649. Kluwer Academic Publishers, Dordrecht (2001)
36. Rubinfeld, R., Sudan, M.: Robust characterizations of polynomials with applications to program testing. SIAM Journal on Computing 25, 252–271 (1996)
37. Silvester, J.R.: Determinants of block matrices. Maths Gazette 84, 460–467 (2000)
38. Smith, H.J.S.: On systems of linear indeterminate equations and congruences. Phil. Trans. Royal Soc. London A 151, 293–326 (1861)
39. Štefankovič, D.: Fourier transform in computer science. Master's thesis, University of Chicago (2000)
40. Terras, A.: Fourier Analysis on Finite Groups and Applications. Cambridge University Press, Cambridge (1999)
41. Yekhanin, S.: Towards 3-query locally decodable codes of subexponential length. Journal of the ACM 55(1), 1–16 (2008)

A Sublogarithmic Approximation for Highway and Tollbooth Pricing*

Iftah Gamzu[1,**] and Danny Segev[2]

[1] Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel
iftgam@tau.ac.il
[2] Department of Statistics, University of Haifa, Haifa 31905, Israel
segevd@stat.haifa.ac.il

Abstract. An instance of the tollbooth problem consists of an undirected network and a collection of single-minded customers, each of which is interested in purchasing a fixed path subject to an individual budget constraint. The objective is to assign a per-unit price to each edge in a way that maximizes the collective revenue obtained from all customers. The revenue generated by any customer is equal to the overall price of the edges in her desired path, when this cost falls within her budget; otherwise, that customer will not purchase any edge.

Our main result is a deterministic algorithm for the tollbooth problem on trees whose approximation ratio is $O(\log m/\log\log m)$, where m denotes the number of edges in the underlying graph. This finding improves on the currently best performance guarantees for trees, due to Elbassioni et al. (SAGT '09), as well as for paths (commonly known as the highway problem), due to Balcan and Blum (EC '06). An additional interesting consequence is a computational separation between tollbooth pricing on trees and the original prototype problem of single-minded unlimited supply pricing, under a plausible hardness hypothesis due to Demaine et al. (SODA '06).

1 Introduction

An extensively-studied question in economics and operations management is that of pricing an assortment of products in a given market, trying to maximize revenue subject to a multitude of constraints. Somewhat informally, the inherent difficulty in such settings boils down to the obvious tension between two extremes: low prices attract more customers, while high prices generate greater revenues per purchase. Recently, the spotlights have been turned on computational challenges in pricing. What seems to be the driving force behind this line

* Due to space limitations, some details are omitted from this extended abstract. All missing details are provided in the full version of this paper [11].

** Supported by the Israel Science Foundation, by the European Commission under the Integrated Project QAP funded by the IST directorate as Contract Number 015848, by a European Research Council (ERC) Starting Grant, and by the Wolfson Family Charitable Trust.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 582–593, 2010.

of research is an immense growth in the range of sources for acquiring costumer preferences data, which are now available as a result of the widespread use of the Internet.

The tollbooth problem. One particular computational task that has received much recent attention is the tollbooth problem, which captures optimization-related aspects of pricing connection links in networks, e.g., setting prices for road use in a system of toll highways. Formally, an instance of this problem consists of an undirected graph $G = (V, E)$ with m edges, which can be broadly interpreted as products with unlimited supply. An additional ingredient of the input is a collection C of n single-minded customers, each of which is interested in purchasing a fixed path subject to an individual budget constraint. Technically speaking, the demand attributes of customer i are represented by a pair (P_i, b_i), where $P_i \subseteq G$ is the path she wishes to buy, and b_i stands for her budget, namely, the maximum price she is willing to pay for that path. Any customer will buy a single unit of each edge in the desired path when its total cost falls within her budget; otherwise, she leaves without buying anything. With this setting in mind, the goal is to assign a per-unit price to each edge in a way that maximizes the overall revenue. More precisely, the objective is to compute a pricing scheme $p : E \to \mathbb{R}_+$ that maximizes the total revenue over all customers, $\sum_{i=1}^{n} R_p(i)$. Here, $R_p(i)$ denotes the revenue obtained from customer i, which evaluates to $\sum_{e \in P_i} p(e)$ when this cost does not exceed b_i, or to 0, otherwise.

Previous work. Guruswami et al. [13] seem to have been the first to study the tollbooth problem. Their main results in this context were to show that tollbooth pricing is APX-hard even when the underlying graph is a tree, and to devise exact dynamic-programming algorithms for the single-source variant on trees and for several other special cases. Additional hardness results were obtained by Briest and Krysta [4], who proved that even the seemingly-manageable setting of a simple path, commonly known as the highway problem, is in fact weakly NP-hard. Elbassioni, Raman, Ray, and Sitters [7] extended this result by establishing strong NP-hardness. On the positive side, however, Balcan and Blum [2] devised an $O(\log m)$ approximation for the highway problem; this finding is incomparable with the quasi-PTAS developed by Elbassioni, Sitters, and Zhang [8] later on. Finally, and very recently, Elbassioni et al. [7] proposed an $O(\log m)$ approximation for arbitrary trees. To conclude, approximating the tollbooth problem on trees, or even on simple paths, beyond the logarithmic threshold has remained an open research question.

1.1 Our Results

The main result of this paper is a deterministic algorithm for the tollbooth problem on trees whose approximation ratio is $O(\log m / \log \log m)$, improving on the currently best performance guarantees for trees, as well as for paths, due to Elbassioni et al. [7], and to Balcan and Blum [2], respectively. Even though the quantitative magnitude of improvement is not that dramatic, our findings have additional interesting contributions:

Conceptual. We identify a computational separation between tollbooth pricing on trees and the original prototype problem of single-minded unlimited supply pricing (see Section 1.2). The latter cannot be approximated within a factor smaller than $\Omega(\log n)$, under a plausible hardness hypothesis regarding the balanced bipartite independent set problem [6], whereas in the former the resulting factor is better by $\Omega(\log \log n)$ or more. Regarding the relation between m and n, we remark that an arbitrary instance of tollbooth pricing on trees can be reduced to one with $m = O(n)$. The general idea is that, when the number of edges is significantly larger than the number of customers, paths can be iteratively contracted (see, for instance, [7, Sec. 2.2]).

Technical. We believe that some of the algorithmic tools and analysis methods, illustrated in Sections 2 and 3, are of independent interest, and may be applicable in other settings as well. In particular, we have already derived new approximability bounds for graph orientation problems [12] by synthesizing ideas such as balanced decompositions, segment guessing, and randomization.

1.2 Related Work

We proceed with a brief discussion on the single-minded unlimited supply pricing problem, from which the computational setting considered in this paper seems to have emerged. The input to the former problem consists of m different products, with unlimited supply, and a collection of n single-minded customers, each of which is interested in purchasing a particular subset (or bundle) of products subject to an individual budget constraint. The goal is to assign a per-unit price to each product in a way that maximizes the overall revenue, where again, a customer will buy a single unit of each product in her bundle only when its total cost fits within her budget. This problem was originally introduced by Guruswami et al. [13], who demonstrated that the single-price policy, where all products are given identical prices, guarantees an approximation ratio of $O(\log n + \log m)$ with respect to the optimal revenue. Later on, Balcan, Blum and Mansour [3] extended this finding to the setting of customers with general valuation functions. From a hardness point of view, Demaine, Feige, Hajiaghayi, and Salavatipour [6] established several inapproximability results under various complexity assumptions. In particular, they proved a lower bound of $\Omega(\log n)$, under a plausible hardness hypothesis regarding the balanced bipartite independent set problem.

A concurrent line of research focused on computing approximate pricing schemes in terms of other problem parameters. For instance, when the number of different products is fixed, Hartline and Koltun [14] showed that an FPTAS can be devised. In addition, Briest and Krysta [4] suggested an $O(\log B + \log \ell)$ approximation, where B is the maximum number of requests per product and ℓ is the maximum size of any bundle, as well as a different algorithm, whose performance guarantee is $O(\ell^2)$. Balcan and Blum [2] improved on this result, to obtain a ratio of $O(\ell)$, and demonstrated that the vertex pricing problem (where $\ell = 2$) can be approximated within a factor of 4. Vertex pricing was further studied in [17,16]. Recently, Cheung and Swamy [5] design an LP-based algorithm for

a more general model of revenue maximization in a limited supply scenario that implies an $O(\log B)$ approximation for single-minded unlimited supply pricing.

2 The Classification Process

Prior to describing the specifics of our approach in detail, which will inevitably involve delving into technicalities, it would be instructive to concentrate on the bigger picture. For this purpose, we begin by pointing out that the performance guarantee of $O(\log m/\log\log m)$ is obtained by employing the classify-and-select paradigm. More specifically, we exploit various structural properties to partition the collection of customers into $O(\log m/\log\log m)$ pairwise-disjoint classes. For each such class, given the additional structure imposed, we separately compute a pricing scheme whose overall revenue comes within a *constant factor* of the optimal revenue attainable from this class. In particular, each class is treated in a completely independent fashion, as if there are no other classes under consideration. Consequently, since the objective function is subadditive, the above-mentioned approximation ratio follows by picking, out of the set of all pricing schemes computed, the one that collects maximal revenue. We proceed by describing the customer classification process; this exposition will allow us to focus attention on the more involved single-class problem later on.

2.1 Classifying Customers via Balanced Decompositions

In the following, we give a formal account of the process by which customers are partitioned into classes. To this end, we begin by introducing the notion of an almost-balanced decomposition, which can be viewed as a generalization of the well-known centroid decomposition [10]. Note that structural properties in this spirit have been explored and exploited in various settings (see, e.g., [9,15]).

Definition 1. *Let $T = (V, E)$ be a tree. An* almost balanced k-decomposition *of T is a partition of T into k edge-disjoint subtrees $T_1, \ldots, T_k$ such that each subtree contains between $|E|/(3k)$ and $3|E|/k$ edges.*

Lemma 1. *Let $T = (V, E)$ be a tree with $|E| \geq k$. An almost balanced k-decomposition of T exists and can be found in polynomial time. Moreover, the number of vertices that are shared by at least two subtrees is less than k.*

The classification process corresponds to a recursive decomposition of the input tree T; to better understand the upcoming discussion, we advise the reader to consult Figure 1. Let $\mathcal{T}_1 = \{T_1, \ldots, T_k\}$ be an almost balanced k-decomposition of T into k edge-disjoint subtrees. The first class of customers, C_1, consists of all customers separated by $\mathcal{T}_1$, that is, customers i for which the endpoints of the desired path P_i (henceforth, s_i and t_i) reside in different subtrees of the decomposition $\mathcal{T}_1$. Now, to classify the remaining set of customers, $C \setminus C_1$, we recursively apply the previously-described procedure with respect to the collection of subtrees in $\mathcal{T}_1$. Specifically, in the second level of the recursion, an almost

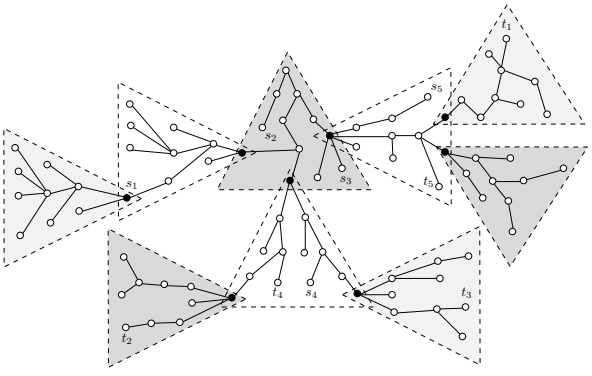


Fig. 1. A schematic example for the collection of customers separated by an almost balanced k-decomposition, where each triangle marks a single subtree. Here, customers 1, 2, and 3 are separated, whereas 4 and 5 are not.

balanced k-decomposition is computed in each of the subtrees $T_1, \ldots, T_k$, to obtain a set $\mathcal{T}_2$, comprising of k^2 subtrees. The second class of customers, C_2, consists of all yet-unclassified customers separated by $\mathcal{T}_2$. In other words, the endpoints of each path P_i, for which $i \in C_2$, reside in different subtrees of $\mathcal{T}_2$, but in the same subtree of $\mathcal{T}_1$. The remaining classes $C_3, C_4, \ldots$ are defined similarly. It is important to note that the recursive process ends as soon as we arrive at a subtree with strictly less than k edges. In this case, we make use of the trivial decomposition, where the given subtree is broken into individual edges.

It is quite obvious that, up until this point in time, k was treated as a parameter whose value has not been determined yet. For our purposes, we employ the above-mentioned classification process with $k = \lceil \log^{1/2} m \rceil$. With this value of k at hand, one can easily verify that the overall number of levels in the recursion, or equivalently, the number of customer classes is $O(\log_k m) = O(\log m / \log\log m)$. This claim is immediately implied by observing that the maximum size of a subtree in level ℓ of the recursion is at most $(3/k)^\ell \cdot |E|$. As a side note, we remark that the balanced decomposition property, ensuring that all subtrees are of size roughly $|E|/k$, does not play any role from this point on, as its sole purpose was to restrict the number of classes to $O(\log m / \log\log m)$.

2.2 Why Handling a Single Decomposition Is Sufficient

We remind the reader that a class of customers, say C_ℓ, generally consists of several subsets of customers, each created when different subtrees in $\mathcal{T}_{\ell-1}$ are partitioned by the decomposition $\mathcal{T}_\ell$. More specifically, assuming that the subtrees in $\mathcal{T}_{\ell-1}$ are $T_1, T_2, \ldots$, the class C_ℓ can be written as the disjoint union of $C_\ell^1, C_\ell^2, \ldots$, where C_ℓ^j is the set of customers that are first separated when T_j is partitioned. Notice that the desired path of any customer separated by some

subtree decomposition must be contained in that subtree, since otherwise, this customer would have been separated in previous recursion steps. This observation implies that it is sufficient to compute an approximate pricing scheme for a single subtree decomposition and its induced set of separated customers. Given a polynomial-time algorithm that computes such pricing schemes, one can *sequentially* apply it to each of the subtree decompositions in the same recursion level. The resulting pricing schemes (in edge-disjoint subtrees) can then be "glued" to form a single scheme, defined for the entire edge set, collecting at least as much revenue as the sum of all individual subtree revenues.

3 The Single Decomposition Algorithm

In what follows, we focus our attention on a single decomposition, and devise a randomized algorithm that computes a pricing scheme whose expected revenue is within a constant factor of the optimal revenue attainable for this decomposition. Later on, we will argue that this algorithm can be easily derandomized. Formally, an instance of the problem in question consists of a tree $T = (V, E)$, and a partition $\mathcal{T} = \{T_1, \ldots, T_k\}$ of this tree into k edge-disjoint subtrees, where $k \leq \lceil \log^{1/2} m \rceil$, such that the number of vertices shared by at least two subtrees is less than k. In addition, we are given a collection C of n customers, satisfying the following properties:

1. Each customer i wishes to purchase a path P_i so long as the overall price of this path does not exceed the budget b_i.
2. Each customer path P_i is separated by $\mathcal{T}$, meaning that the endpoints of P_i reside in different subtrees of the decomposition $\mathcal{T}$.

3.1 Notation and Terminology

For ease of presentation, it would be convenient to introduce some notation and terminology before describing the nuts and bolts of our algorithm. To better understand the suggested notation, we refer the reader to the example in Figure 2.

- Let $p^* : E \to \mathbb{R}_+$ be an optimal pricing scheme, with a revenue of OPT.
- Let $V_B \subseteq V$ be the set of *border vertices* of $\mathcal{T}$, that is, the set of vertices that are shared by at least two subtrees in $\mathcal{T}$. In addition, let $\mathcal{S} \subseteq T$ be the *skeleton* of $\mathcal{T}$, namely, the minimal subtree spanned by all border vertices. Note that this subtree consists of the union of paths connecting any two vertices in V_B.
- Now, recall that the endpoints of each customer path P_i reside in different subtrees of the decomposition $\mathcal{T}$, meaning that P_i must traverse at least one border vertex. Therefore, we can divide each customer path, with endpoints s_i and t_i, into three (possibly empty) parts:
  1. A subpath between s_i and its closest skeleton vertex v_{s_i}.
  2. A subpath between t_i and its closest skeleton vertex v_{t_i}.
  3. A subpath between v_{s_i} and v_{t_i}, along the skeleton.

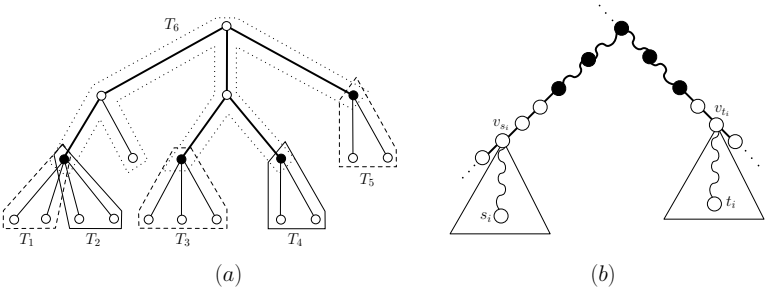


Fig. 2. (a) An almost balanced 6-decomposition of a tree with 18 edges. Note that the black vertices are border vertices, and the heavy edges make up the skeleton $\mathcal{S}$. (b) Dividing the customer path P_i into three parts.

Based on this definition, let $R^S_{p^*}(i)$, $R^T_{p^*}(i)$, and $R^M_{p^*}(i)$ denote the revenues obtained in the optimal pricing scheme p^* from the subpaths of customer i, respectively[1]. Clearly, $R_{p^*}(i) = R^S_{p^*}(i) + R^T_{p^*}(i) + R^M_{p^*}(i)$.

3.2 The Algorithm

Having all necessary definitions in place, we are now ready to present the specifics of our pricing algorithm, and to analyze its performance. In the remainder of this section, we consider two complementing scenarios, depending on the contribution of different path parts to the revenue generated by the optimal pricing scheme p^*. Somewhat informally, the first scenario captures a situation where a significant portion of OPT is gained from customer subpaths traversing non-skeleton edges, or in other words, when $\sum_{i=1}^{n}(R^S_{p^*}(i)+R^T_{p^*}(i)) = \Omega(1)\cdot\text{OPT}$. The second scenario corresponds to a situation where a large portion is delivered by subpaths along the skeleton, that is, when $\sum_{i=1}^{n} R^M_{p^*}(i) = \Omega(1)\cdot\text{OPT}$. For each of these scenarios, we compute a pricing scheme whose expected revenue is within a constant factor of optimal. As a result, we obtain a constant approximation ratio by computing both pricing schemes and picking the one that achieves the maximum revenue. For sake of simplicity, we begin by considering the easy-to-handle first scenario, noting that the second, more challenging scenario, will be discussed in the sequel.

Scenario I: $\sum_{i=1}^{n}(R^S_{p^*}(i) + R^T_{p^*}(i)) \geq \text{OPT}/2$

In the present setting, at least half of the optimal revenue is collected from customer subpaths consisting of non-skeleton edges, meaning that the collective contribution of the subpaths $P_i \setminus \mathcal{S}$, over all customers i, is at least OPT/2. The algorithmic tool that allows us to handle this scenario is a polynomial-time procedure, due to Guruswami et al. [13], for solving the single-source tollbooth problem (on trees) to optimality. Here, the underlying assumption is that all

[1] These superscripts stand for: S – subpath adjacent to s_i; T – subpath adjacent to t_i; and M – middle subpath.

customer paths share a common endpoint, that is, $s_1 = \cdots = s_n$. With this tool at hand, the algorithm proceeds as follows:

1. Each decomposition subtree T_j is randomly and independently marked as being *active*, with probability 1/2; otherwise, that tree is *inactive*.
2. We decide in advance that all skeleton edges, as well as all edges within inactive trees, could be purchased free of charge, i.e., their price is zero.
3. For each active subtree T_j, let $\mathcal{E}_j$ be the collection of customer path endpoints that reside in T_j; note that each customer contributes at most one endpoint to this collection[2]. We now contract all skeleton edges of T_j into a single root[3], designated by r, and apply the single-source algorithm, when the underlying set of customers are only those with an endpoint in $\mathcal{E}_j$. Each such customer i is now interested in purchasing the path connecting $\{s_i, t_i\} \cap \mathcal{E}_j$ to the root r, still with a budget of b_i.

Lemma 2. *The computed pricing scheme guarantees an expected revenue of at least* OPT/8.

Scenario II: $\sum_{i=1}^{n} R_{p^*}^{M}(i) \geq \mathrm{OPT}/2$

In this setting, at least half of the optimal revenue is collected from customer subpaths consisting of skeleton edges, meaning that the collective contribution of the subpaths $P_i \cap \mathcal{S}$, over all customers i, is at least OPT/2. Our algorithm begins by first deciding that all non-skeleton edges could be purchased for free, and sets their price to zero. As a result, we may assume that the endpoints of each customer path are located on the skeleton, since otherwise, we can relocate them to their closest skeleton vertices, without any consequences whatsoever. With this structural alteration in mind, the skeleton pricing is carried out in two phases: *segment guessing*, where close estimates of the optimal prices along disjoint subpaths of the skeleton are obtained, followed by *randomized assignment*, where prices are associated with individual skeleton edges.

Phase I: segment guessing. We remind the reader that the skeleton $\mathcal{S}$ is the minimal subtree of T spanned by all border vertices V_B. We denote by V_J the set of *junction vertices*, defined as non-border skeleton vertices with degree at least 3 (counting only skeleton edges); it is not difficult to verify that $|V_J| < |V_B| < k$. Finally, we call the vertex set $V_B \cup V_J$ the *core* of the skeleton $\mathcal{S}$. Based on these definitions, we can partition the skeleton into a collection $\Sigma(\mathcal{S})$ of edge-disjoint paths, which are referred to as *segments*. Each such segment is a subpath of $\mathcal{S}$ whose endpoints are core vertices, but its interior traverses only non-core vertices. Obviously, $|\Sigma(\mathcal{S})| = |V_B| + |V_J| - 1 < 2k$.

In what follows, we argue that one could obtain in polynomial time a close estimate for the total price $p^*(\sigma) = \sum_{e \in \sigma} p^*(e)$ of each segment $\sigma \in \Sigma(\mathcal{S})$,

[2] Otherwise, T_j contains her desired path, in contradiction to all customer paths in question being separated by the decomposition $\mathcal{T}$.

[3] That is, skeleton edges are removed, and their endpoints are unified into a representative vertex.

simultaneously for all segments. To this end, it is sufficient to prove that there exists a *very small* set of prices Γ, and a corresponding pricing scheme $p_\Gamma : E \to \mathbb{R}_+$, such that the price given by p_Γ to every segment falls within Γ, i.e., $p_\Gamma(\sigma) \in \Gamma$ for each $\sigma \in \Sigma(\mathcal{S})$, and such that $p_\Gamma(\sigma)$ approximates $p^*(\sigma)$ well. At the same time, we would like to make sure that the overall revenue from customer subpaths consisting of skeleton edges still forms a fixed portion of $\sum_{i=1}^{n} R^M_{p^*}(i)$. A construction of this nature is given in the next lemma.

Lemma 3. *Let* $\Gamma = \{2^\ell \cdot b_{\max}/(4nm) : 0 \leq \ell \leq \lfloor \log(4nm^2) \rfloor\} \cup \{0\}$, *where* $b_{\max} = \max_i b_i$. *There is a pricing scheme* $p_\Gamma : E \to \mathbb{R}_+$ *that satisfies* (i) $p_\Gamma(\sigma) \in \Gamma$ *for each* $\sigma \in \Sigma(\mathcal{S})$, *and* (ii) $\sum_{i=1}^{n} R^M_{p_\Gamma}(i) \geq \sum_{i=1}^{n} R^M_{p^*}(i)/4$.

By Lemma 3, we conclude that to obtain a close estimate for the total price $p^*(\sigma)$ of each segment $\sigma \in \Sigma(\mathcal{S})$, simultaneously for all segments, the total number of price assignments to be examined is of polynomial size, since

$$|\Gamma|^{|\Sigma(\mathcal{S})|} = \big(O(\log(nm))\big)^{O(k)} = \big(O(\log(nm))\big)^{O(\log^{1/2} m)} = o(nm) \ .$$

Consequently, we may assume without loss of generality that the collection of segment prices $\{p_\Gamma(\sigma) : \sigma \in \Sigma(\mathcal{S})\}$, given in Lemma 3, is known in advance. This assumption can be easily enforced by enumerating over all $o(nm)$ possible assignments. On the other hand, it is worth noting that we do not assume any knowledge of the edge-specific pricing $p_\Gamma : E \to \mathbb{R}_+$.

Phase II: randomized assignment. The goal of this phase is to complete the pricing scheme by assigning carefully-picked random prices to individual skeleton edges, based on the estimated segment prices $\{p_\Gamma(\sigma) : \sigma \in \Sigma(\mathcal{S})\}$. The crux lies in making sure that the edge-specific prices *always* respect the outcome of the segment guessing phase, as stated in the next invariant.

Invariant 1. *With probability* 1, *the total price of each segment* $\sigma \in \Sigma(\mathcal{S})$ *is exactly* $p_\Gamma(\sigma)$.

Our assignment procedure consists of independent steps, where segments are processed one after the other. In each step, we consider a skeleton segment $\sigma = \langle v_1, \ldots, v_\ell \rangle$, and assign prices to its edges $(v_1, v_2), \ldots, (v_{\ell-1}, v_\ell)$ in a way that satisfies Invariant 1. Specifically, as illustrated in Figure 3(a), we pick one of the following four price assignments uniformly at random:

1. Assign a price of $p_\Gamma(\sigma)$ to (v_1, v_2), and zero prices to the remaining edges.
2. Assign a price of $p_\Gamma(\sigma)$ to $(v_{\ell-1}, v_\ell)$, and zero prices to the remaining edges.
3. Assign prices based on a v_1-rooted single-source problem (see below).
4. Assign prices based on a v_ℓ-rooted single-source problem (similar to item 3).

To complete the description of our algorithm, it remains to explain how the single-source instances (in items 3 and 4) are created and solved; for brevity of presentation, we focus on the v_1-rooted case, noting that the opposite case is identical, up to changing the roles of v_1 and v_ℓ. Once again, we will employ the dynamic-programming algorithm of Guruswami et al. [13] for solving the single-source tollbooth problem on trees, a tool that was introduced in Scenario I. In particular, the v_1-rooted instance is comprised of the following components:

- The underlying graph is the segment σ.
- The set of customers are those with an endpoint in $\{v_2, \ldots, v_{\ell-1}\}$, whose desired path exits the segment σ through v_1.
- For each customer i under consideration, we set up a new endpoint at v_1, instead of the one outside σ, and change her budget to $\min\{p_\Gamma(\sigma), b_i - \sum_{\bar{\sigma}:\bar{\sigma}\subseteq P_i} p_\Gamma(\bar{\sigma})\}$. Note that the latter term is exactly the budget remaining to purchase P_i, assuming that a total price of $p_\Gamma(\bar{\sigma})$ has already been paid for each segment $\bar{\sigma} \in \Sigma(\mathcal{S})$ that is fully-contained in P_i.

Needless to say, the single-source algorithm does not set a price for the edge $(v_{\ell-1}, v_\ell)$, as it is not contained in any desired path of the v_1-rooted instance. Hence, to ensure that Invariant 1 holds, we set the price of $(v_{\ell-1}, v_\ell)$ to be the difference between the total segment price $p_\Gamma(\sigma)$ and the total newly-computed price of $(v_1, v_2), \ldots, (v_{\ell-2}, v_{\ell-1})$. Here, it is important to point out that this difference is indeed non-negative, since the dynamic-programming algorithm guarantees that the price of each subpath $\langle v_1, \ldots, v_j\rangle$ is equal to the budget of some customer whose desired path is contained in $\langle v_1, \ldots, v_j\rangle$ (see [13, Thm. 5.3]); on the other hand, the budget of every customer cannot exceed $p_\Gamma(\sigma)$, by definition.

Analysis. The remainder of this section is devoted to proving that the expected revenue of the pricing scheme computed in the randomized assignment phase is within a constant factor of optimal, as formally stated in the following lemma.

Lemma 4. *The pricing scheme constructed in the randomized assignment phase guarantees an expected revenue of at least* OPT/256.

Recall that we have previously assumed the endpoints of each customer path P_i to be located on the skeleton. Moreover, since these endpoints reside in different subtrees of the decomposition $\mathcal{T}$, the path P_i must traverse at least one border vertex. For this reason, as shown in Figure 3(b), we can divide each customer path, with endpoints s_i and t_i, into three (possibly empty) parts:

1. A subpath, along partial segment, between s_i and its closest core vertex u_{s_i}.
2. A subpath, along partial segment, between t_i and its closest core vertex u_{t_i}.
3. A subpath between u_{s_i} and u_{t_i}, along a sequence of complete segments.

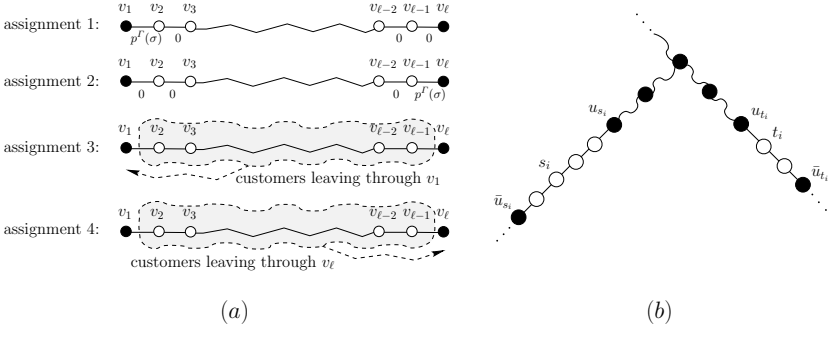


Fig. 3. (a) A schematic description of the four random assignments for the segment σ. Here, core vertices are marked in black. (b) The new customer path partition.

With these definitions in mind, let $R^{\tilde{S}}_{p_\Gamma}(i)$, $R^{\tilde{T}}_{p_\Gamma}(i)$, and $R^{\tilde{M}}_{p_\Gamma}(i)$ be the revenues obtained in the estimated pricing scheme p_Γ from the subpaths of customer i, respectively. We remark that p_Γ is the pricing scheme whose existence has been established in Lemma 3. Note that, by item (ii) of this lemma, we can bound the sum of the above-mentioned revenues, over all customers, in terms of OPT, since

$$\sum_{i=1}^{n} \left(R^{\tilde{S}}_{p_\Gamma}(i) + R^{\tilde{T}}_{p_\Gamma}(i) + R^{\tilde{M}}_{p_\Gamma}(i) \right) = \sum_{i=1}^{n} R^{M}_{p_\Gamma}(i) \geq \frac{1}{4} \sum_{i=1}^{n} R^{M}_{p^*}(i) \geq \frac{\mathrm{OPT}}{8} \,.$$

In what follows, we consider two cases, depending on the contribution of different path parts to the revenue generated by the pricing scheme p_Γ. For each of these cases, we show that the randomized assignment phase computes a pricing scheme whose expected revenue is within a constant factor of optimal.

Case I: $\sum_{i=1}^{n} R^{\tilde{M}}_{p_\Gamma}(i) \geq \mathrm{OPT}/16$. In this setting, a significant fraction of the optimal revenue is collected from customer subpaths consisting of complete segments, meaning that the collective contribution of the paths $u_{s_i} \rightsquigarrow u_{t_i}$, over all customers i, is at least OPT/16. Notice that, by Invariant 1, our pricing scheme always assigns a total price of $p_\Gamma(\sigma)$ to each segment $\sigma \in \Sigma(\mathcal{S})$, implying that the overall price of $u_{s_i} \rightsquigarrow u_{t_i}$ is exactly $R^{\tilde{M}}_{p_\Gamma}(i)$. Therefore, the revenue from customer i is at least $R^{\tilde{M}}_{p_\Gamma}(i)$, unless the total price of the remaining partial segments $s_i \rightsquigarrow u_{s_i}$ and $t_i \rightsquigarrow u_{t_i}$ exceeds the residual budget, $b_i - R^{\tilde{M}}_{p_\Gamma}(i)$. We next argue that, with probability at least 1/16, all edges on these partial segments will be given zero prices. Consequently, the expected revenue from customer i is at least $R^{\tilde{M}}_{p_\Gamma}(i)/16$, and by linearity of expectation, the overall expected revenue is no less than $\sum_{i=1}^{n} R^{\tilde{M}}_{p_\Gamma}(i)/16 \geq \mathrm{OPT}/256$.

For the purpose of establishing the previously mentioned claim, we will show that, with probability at least 1/4, each and every edge on $s_i \rightsquigarrow u_{s_i}$ is assigned a zero price. The claim then follows by observing that an identical property with respect to $t_i \rightsquigarrow u_{t_i}$ can be proven along the same lines, and also, that the two events are independent. Now, notice that if s_i is a core vertex then the claim trivially holds, since $s_i \rightsquigarrow u_{s_i}$ is empty. Hence, let us consider the case where s_i is an internal vertex on the segment between the core vertices u_{s_i} and $\bar{u}_{s_i}$. In this case, with probability 1/4, our algorithm assigns a price of zero to all edges in this segment, except for the edge adjacent to $\bar{u}_{s_i}$; in particular, all edges on $s_i \rightsquigarrow u_{s_i}$ are assigned zero prices.

Case II: $\sum_{i=1}^{n} (R^{\tilde{S}}_{p_\Gamma}(i) + R^{\tilde{T}}_{p_\Gamma}(i)) \geq \mathrm{OPT}/16$. The analysis here is somewhat more involved, and we provide further details in [11].

3.3 Derandomization

The avid reader may already have noticed that the extent to which we utilize randomization is rather limited. More specifically, in Scenario I each subtree in the decomposition is randomly marked as being active or inactive, whereas in Scenario II one of four possible price assignments is picked at random for each

segment. In other words, all we need to obtain a deterministic algorithm are two uniform sample spaces, with $O(1)$ possible values for $O(\log^{1/2} m)$ random variables. These can be constructed in polynomial time either explicitly, as there are only $O(m^{O(1)})$ outcomes to examine, or in a more compact way, by observing that nothing more than pairwise-independence (see, for instance, [1, Chap. 15]) is required for the preceding analysis.

References

1. Alon, N., Spencer, J.H.: The Probabilistic Method, 2nd edn. Wiley, New York (2000)
2. Balcan, M.F., Blum, A.: Approximation algorithms and online mechanisms for item pricing. Theory of Computing 3(1), 179–195 (2007)
3. Balcan, M.F., Blum, A., Mansour, Y.: Item pricing for revenue maximization. In: EC, pp. 50–59 (2008)
4. Briest, P., Krysta, P.: Single-minded unlimited supply pricing on sparse instances. In: SODA, pp. 1093–1102 (2006)
5. Cheung, M., Swamy, C.: Approximation algorithms for single-minded envy-free profit-maximization problems with limited supply. In: FOCS, pp. 35–44 (2008)
6. Demaine, E.D., Feige, U., Hajiaghayi, M., Salavatipour, M.R.: Combination can be hard: Approximability of the unique coverage problem. SIAM Journal of Computing 38(4), 1464–1483 (2008)
7. Elbassioni, K.M., Raman, R., Ray, S., Sitters, R.: On profit-maximizing pricing for the highway and tollbooth problems. In: Mavronicolas, M., Papadopoulou, V.G. (eds.) SAGT 2009. LNCS, vol. 5814, pp. 275–286. Springer, Heidelberg (2009)
8. Elbassioni, K.M., Sitters, R., Zhang, Y.: A quasi-PTAS for profit-maximizing pricing on line graphs. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA 2007. LNCS, vol. 4698, pp. 451–462. Springer, Heidelberg (2007)
9. Even, G., Garg, N., Könemann, J., Ravi, R., Sinha, A.: Covering graphs using trees and stars. In: Arora, S., Jansen, K., Rolim, J.D.P., Sahai, A. (eds.) RANDOM 2003 and APPROX 2003. LNCS, vol. 2764, pp. 24–35. Springer, Heidelberg (2003)
10. Frederickson, G.N., Johnson, D.B.: Generating and searching sets induced by networks. In: de Bakker, J.W., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 221–233. Springer, Heidelberg (1980)
11. Gamzu, I., Segev, D.: A sublogarithmic approximation for highway and tollbooth pricing, http://arxiv.org/abs/1002.2084
12. Gamzu, I., Segev, D.: An improved approximation algoritm for maximum graph orientation (2010) (in preparation)
13. Guruswami, V., Hartline, J.D., Karlin, A.R., Kempe, D., Kenyon, C., McSherry, F.: On profit-maximizing envy-free pricing. In: SODA, pp. 1164–1173 (2005)
14. Hartline, J.D., Koltun, V.: Near-optimal pricing in near-linear time. In: Dehne, F., López-Ortiz, A., Sack, J.-R. (eds.) WADS 2005. LNCS, vol. 3608, pp. 422–431. Springer, Heidelberg (2005)
15. Hassin, R., Segev, D.: Robust subgraphs for trees and paths. ACM Transactions on Algorithms 2(2), 263–281 (2006)
16. Khandekar, R., Kimbrel, T., Makarychev, K., Sviridenko, M.: On hardness of pricing items for single-minded bidders. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX 2009. LNCS, vol. 5687, pp. 202–216. Springer, Heidelberg (2009)
17. Krauthgamer, R., Mehta, A., Rudra, A.: Pricing commodities, or how to sell when buyers have restricted valuations. In: Kaklamanis, C., Skutella, M. (eds.) WAOA 2007. LNCS, vol. 4927, pp. 1–14. Springer, Heidelberg (2008)

Maximum Quadratic Assignment Problem: Reduction from Maximum Label Cover and LP-Based Approximation Algorithm

Konstantin Makarychev[1], Rajsekar Manokaran[2,⋆], and Maxim Sviridenko[1]

[1] IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598
[2] Princeton University, Princeton NJ 08544

Abstract. We show that for every positive $\varepsilon > 0$, unless $\mathcal{NP} \subset \mathcal{BPQP}$, it is impossible to approximate the maximum quadratic assignment problem within a factor better than $2^{\log^{1-\varepsilon} n}$ by a reduction from the maximum label cover problem. Then, we present an $O(\sqrt{n})$-approximation algorithm for the problem based on rounding of the linear programming relaxation often used in the state of the art exact algorithms.

1 Introduction

In this paper we consider the Quadratic Assignment Problem. An instance of the problem, $\Gamma = (G, H)$ is specified by two weighted graphs $G = (V_G, w_G)$ and $H = (V_H, w_H)$ such that $|V_G| = |V_H|$ (we denote $n = |V_G|$). The set of feasible solutions consists of bijections from V_G to V_H. For a given bijection φ the objective function is

$$\text{value}_{\text{QAP}}(\Gamma, \varphi) = \sum_{(u,v) \in V_G \times V_G} w_G(u,v) w_H(\varphi(u), \varphi(v)). \tag{1}$$

There are two variants of the problem the Minimum Quadratic Assignment Problem and the Maximum Quadratic Assignment Problem (MaxQAP) where the objective function (1) should be minimized or maximized respectively. The problem was first defined by Koopmans and Beckman [21] and sometimes this formulation of the problem is referred to as Koopmans-Beckman formulation of the Quadratic Assignment Problem. Both variants of the problem model an astonishingly large number of combinatorial optimization problems such as traveling salesman, maximum acyclic subgraph, densest subgraph and clustering problems to name a few. It also generalizes many practical problems that arise in various areas such as modeling of backboard wiring [29], campus and hospital layout [13,15], scheduling [18] and many others [14,23]. The surveys and books [2,9,11,10,24,25] contain an in-depth treatment of special cases and various applications of the Quadratic Assignment Problem.

⋆ Supported by NSF Grants CCF-0832797, 0830673, and 0528414.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 594–604, 2010.

The Quadratic Assignment Problem is an extremely difficult optimization problem. The state of the art exact algorithms can solve instances with approximately 30 vertices, so a lot of research effort was concentrated on constructing good heuristics and relaxations of the problem.

Previous Results. The Minimum Quadratic Assignment Problem is known to be hard to approximate even under some very restrictive conditions on the weights of graphs G and H [28,19]. Polynomial time exact [11] and approximation algorithms [19] are known for very specialized instances.

In contrast, MaxQAP seem to be more tractable: $O(\sqrt{n}\log^2 n)$-approximation algorithm was constructed by Nagarajan and Sviridenko [22] by utilizing approximation algorithms for the minimum vertex cover, densest k-subgraph and star packing problems. For the special case when one of the edge weight functions (w_G or w_H) satisfy the triangle inequality there are combinatorial 4-approximation [3] and LP-based 3.16-approximation algorithms [22]. Another tractable special case is the so-called dense Quadratic Assignment Problem [4] this special case admits a sub-exponential time approximation scheme and in some cases it could be implemented in polynomial time [4,17]. On the negative side, APX-hardness of MaxQAP is implied by the APX-hardness of its special cases, e.g. Traveling Salesman Problem with Distances One and Two [26].

An interesting special case of MaxQAP is the Densest k-Subgraph Problem. The best known algorithm by Bhaskara, Charikar, Chlamtac, Feige, and Vijayaraghavan [8] gives a $O(n^{1/4})$ approximation. However, the problem is not even known to be APX-hard (under standard complexity assumptions). Feige [16] showed that the Densest k-Subgraph Problem does not admit a ρ-approximation (for some universal constant $\rho > 1$) assuming that random 3-SAT formulas are hard to refute. Khot [20] ruled out PTAS for the problem under the assumption that $\mathcal{NP}$ does not have randomized algorithms that run in sub-exponential time.

Our Results. Our first result is the first superconstant non-approximability for MaxQAP. We show that for every positive $\varepsilon > 0$, unless $\mathcal{NP} \subset \mathcal{BPQP}$ ($\mathcal{BPQP}$ is the class of problems solvable in randomized quasi-polynomial time), it is impossible to approximate the maximum quadratic assignment problem with the approximation factor better than $2^{\log^{1-\varepsilon} n}$. Particularly, there is no polynomial time poly-logarithmic approximation algorithms for MaxQAP under the above complexity assumption. It is an interesting open question if our techniques can be used to obtain a similar result for the Densest k-Subgraph Problem.

Our second result is an $O(\sqrt{n})$-approximation algorithm based on rounding of the optimal solution of the linear programming relaxation. The LP relaxation was first considered by Adams and Johnson [1] in 1994. As a consequence of our result we obtain a bound of $O(\sqrt{n})$ on the integrality gap of this relaxation that almost matches a lower bound of $\Omega(\sqrt{n}/\log n)$ of Nagarajan and Sviridenko [22]. Note, that the previous $O(\sqrt{n}\log^2 n)$-approximation algorithm [22] was not based on the linear programming relaxation, and therefore no non-trivial upper bound on the integrality gap of the LP was known.

2 Preliminaries

A weighted graph $G = (V, w)$ is specified by a vertex set V along with a weight function $w : V \times V \to \mathbb{R}$ such that for every $u, v \in V$, $w(u, v) = w(v, u)$ and $w(u, v) \geq 0$. An edge $e = (u, v)$ is said to be present in the graph G if $w(u, v)$ is non-zero.

We prove the inapproximability of the MaxQAP problem via an approximation preserving poly-time randomized reduction from the Label Cover problem.

Definition 1 (Label Cover Problem).
An instance of the label cover problem represented as $\Upsilon = (G = (V_G, E_G), \pi, [k])$ *consists of a graph* G *on* V_G *with edge set* E_G *along with a set of labels* $[k] = \{0, 1, \ldots k-1\}$. *For each edge* $(u, v) \in E_G$, *there is a constraint* π_{uv}, *a subset of* $[k] \times [k]$ *defining the set of accepted labelings for the end points of the edge. The goal is to find a labeling of the vertices,* $\Lambda : V_G \to [k]$ *maximizing the total fraction of the edge constraints satisfied. We will denote the optimum of a instance* Υ *by* $\mathrm{OPT}_{LC}(\Upsilon)$. *In other words,*

$$\mathrm{OPT}_{LC}(\Upsilon) \stackrel{\text{def}}{=} \max_{\Lambda: V_G \to [k]} \frac{1}{|E_G|} \sum_{(u,v) \in E} I((\Lambda(u), \Lambda(v)) \in \pi_{uv})$$

We will denote the fraction of edges satisfied by a labeling Λ *by* $\mathrm{value}_{LC}(\Upsilon, \Lambda)$.

3 Hardness of Approximation

The PCP theorem [6,7], along with the Raz parallel repetition theorem [27] shows that the label cover problem is hard to approximate within a factor of $2^{\log^{1-\varepsilon} n}$.

Theorem 1 (see e.g., Arora and Lund [5]). *If* $\mathcal{NP} \not\subset \mathcal{QP}$, *then for every positive* $\varepsilon > 0$, *it is not possible to distinguish satisfiable instances of the label cover problem from instances with optimum at most* $2^{-\log^{1-\varepsilon} n}$ *in polynomial time.*

We will show an approximation preserving reduction from a label cover instance to a MaxQAP instance such that: if the label cover instance Υ is completely satisfiable, the MaxQAP instance Γ with have optimum 1; on the other hand, if $\mathrm{OPT}_{\mathrm{LC}}(\Upsilon)$ is at most δ, then no bijection φ obtains a value greater than $O(\delta)$.

Strictly speaking, the problem is not well defined when the graphs G and H do not have the same number of vertices. However, in our reduction, we will relax this condition by letting G have fewer vertices than H, and allowing the map φ to be only injective (i.e., $\varphi(u) \neq \varphi(v)$, for $u \neq v$). The reason is that we can always add enough isolated vertices to G to satisfy $|V_G| = |V_H|$. We also assume that the graphs are unweighted, and thus given an instance Γ consisting of two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, the goal is to find an injective map $\varphi : V_G \to V_H$, so as maximize

$$\text{value}_{\text{QAP}}(\Gamma, \varphi) = \sum_{(u,v)\in E_G} I((\varphi(u), \varphi(v)) \in E_H),$$

here $I(\cdot)$ denotes the indicator function. We denote the optimum by $\text{OPT}_{\text{QAP}}(\Gamma)$.

Informally, our reduction does the following. Given an instance $\Upsilon = (G = (V_G, E_G), \pi, [k])$ of the label cover problem, consider the *label extended* graph H on $V_G \times [k]$ with edges $((u,i)-(v,j))$ for every $(u,v) \in E_G$ and every accepting label pair $(i,j) \in \pi_{uv}$. Every labeling Λ for Υ naturally defines a injective map, φ between V_G and $V_G \times [k]$: $\varphi(u) = (u, \Lambda(u))$. Note that φ maps edges satisfied by Λ onto edges of H. Conversely, given a injection $\varphi : V_G \to V_G \times [k]$ such that $\varphi(u) \in \{u\} \times [k]$ for every $u \in V_G$, we can construct a labeling Λ for Υ satisfying exactly the constraint edges in G which were mapped on to edges of H. However, the additional restriction on the injection is crucial for the converse to hold: an arbitrary injective map might not correspond to any labeling of the label cover Υ.

To overcome the above shortcoming, we modify the graphs G and H as follows. We replace each vertex u in G with a "cloud" of vertices $\{(u,i) : i \in [N]\}$ and each vertex (u,x) in H with a cloud of vertices $\{(u,x,i) : i \in [N]\}$, each index i is from a significantly large set $[N]$. Call the new graphs $\widetilde{G}$ and $\widetilde{H}$ respectively.

For every edge $(u,v) \in E_G$, the corresponding clouds in $\widetilde{G}$ are connected by a random bipartite graph where each edge occurs with probability α. We do this independently for each edge in E_G. For every accepting pair $(x,y) \in \pi_{uv}$, we copy the "pattern" between the clouds $(u,x,\star)$ and $(v,y,\star)$ in $\widetilde{H}$.

Note, that every solution of the label cover problem $u \mapsto \Lambda(u)$ corresponds to the map $(u,i) \mapsto (u, \Lambda(u), i)$ which maps every "satisfied" edge of $\tilde{G}$ to an edge of $\tilde{H}$. The key observation now is that, we may assume that every (u,i) is mapped to some (u,x,i), since, loosely speaking, the pattern of edges between $(u,\star)$ and $(v,\star)$ is unique for each edge (u,v): there is no way to map the *cloud* of u to the *cloud* of u' and the *cloud* of v to the *cloud* of v' (unless $u = u'$ and $v = v'$), so that more than an α fraction of the edges of one cloud are mapped on edges of the other cloud. We will make the above discussion formal in the rest of this section.

Hardness Reduction

Input: A label cover instance $\Upsilon = (G = (V_G, E_G), \pi, [k])$.

Output: A MaxQAP instance $\Gamma = (\widetilde{G}, \widetilde{H})$; $\widetilde{G} = (V_{\widetilde{G}}, E_{\widetilde{G}})$, $\widetilde{H} = (V_{\widetilde{H}}, E_{\widetilde{H}})$.

Parameters: Let N be an integer bigger than $n^4|E_G|k^5$ and $\alpha = 1/n$.

- Define $V_{\widetilde{G}} = V_G \times [N]$ and $V_{\widetilde{H}} = V_G \times [k] \times [N]$.
- For every edge (u,v) of G pick a random set of pairs $\mathcal{E}_{uv} \subset [N] \times [N]$. Each pair $(i,j) \in [N] \times [N]$ belongs to $\mathcal{E}_{uv}$ with probability α. The probabilities are chosen independently of each other.
- For every edge (u,v) of G and every pair (i,j) in $\mathcal{E}_{uv}$, add an edge $((u,i), (v,j))$ to $\widetilde{G}$. Then

$$E_{\widetilde{G}} = \{((u,i),(v,j)) : (u,v) \in E_G \text{ and } (i,j) \in \mathcal{E}_{uv}\}.$$

– For every edge (u, v) of G, every pair (i, j) in $\mathcal{E}_{uv}$, and every pair (x, y) in π_{uv}, add an edge $((u, x, i), (v, y, j))$ to $\widetilde{H}$. Then

$$E_{\widetilde{H}} = \{((u, x, i), (v, y, j)) : (u, v) \in E_G,\ (i, j) \in \mathcal{E}_{uv} \text{ and } (x, y) \in \pi_{uv}\}.$$

It is easy to see that the reduction runs in polynomial time. The size of the instance produced is nN^2k. In our reduction, both k and N are polynomial in n.

We will now show that the reduction is in fact approximation preserving with high probability. In the rest of the section, we will assume $\Gamma = (\widetilde{G}, \widetilde{H})$ is a MaxQAP instance obtained from a label cover instance Υ using the above reduction with parameteres N and α. Note that Γ is a random variable.

We will first show that if the label cover instance has a good labeling, the MaxQAP instance output by the above reduction has a large optimum. The following claim, which follows from a simple concentration inequality, shows that the graph $\widetilde{G}$ has, in fact, as many edges as expected.

Claim 1. With high probability, $\widetilde{G}$ contains at least $\alpha|E_G|N^2/2$ edges.

Lemma 1 (Completeness). *Let Υ be a satisfiable instance of the Label Cover Problem. Then there exists a map of $\widetilde{G}$ to $\widetilde{H}$ that maps every edge of $\widetilde{G}$ to an edge of $\widetilde{H}$. Thus,* $\mathrm{OPT}_{QAP}(\Gamma) = |E_{\widetilde{G}}|$.

Proof. Let $u \mapsto \Lambda(u)$ be the solution of the label cover that satisfies all constrains. Define the map $\varphi : V_{\widetilde{G}} \to V_{\widetilde{H}}$ as follows $\varphi(u, i) = (u, \Lambda(u), i)$. Suppose that $((u, i), (v, j))$ is an edge in $\widetilde{G}$. Then $(u, v) \in E_G$ and $(i, j) \in \pi_{uv}$. Since the constraint between u and v is satisfied in the instance of the label cover, $(\Lambda(u), \Lambda(v)) \in \pi_{uv}$. Thus, $((u, \Lambda(u), i), (v, \Lambda(v), j)) \in E_{\widetilde{H}}$.

Next, we will bound the optimum of Γ in terms of the value of the label cover instance Υ. We do this in two steps. We will first show that for a fixed map φ from $V_{\widetilde{G}}$ to $V_{\widetilde{H}}$ the expected value of Γ can be bounded as a function of the optimum of Υ. Note that this is well defined as $V_{\widetilde{G}}$ and $V_{\widetilde{H}}$ are determined by Υ and N (and independent of the randomness used by the reduction). Next, we show that the value is, in fact, tightly concentrated around the expected value. Then, we do a simple union bound over all possible φ to obtain the desired result. We prove the statements below in the full version of the paper.

Lemma 2. *For every fixed injective map $\varphi : V_{\widetilde{G}} \to V_{\widetilde{H}}$,*

$$\Pr\left\{\mathrm{value}_{QAP}(\Gamma, \varphi) - \mathbb{E}\left[\mathrm{value}_{QAP}(\Gamma, \varphi)\right] \geq \alpha N^2\right\} \leq e^{-n^2 Nk}.$$

Corollary 1 (Soundness). *With high probability, the reduction outputs an instance Γ such that*

$$\mathrm{OPT}_{QAP}(\Gamma) \leq \alpha|E_G|N^2 \times (\mathrm{OPT}_{LC}(\Upsilon) + 2\alpha)$$

Theorem 2. *For every positive $\varepsilon > 0$, there is no polynomial time approximation algorithm for the Maximum Quadratic Assignment problem with the approximation factor less than $D = 2^{\log^{1-\varepsilon} n}$ (where n is the number of vertices in the graph) unless $\mathcal{NP} \subset \mathcal{BPQP}$.*

4 LP Relaxation and Approximation Algorithm

We now present a new $O(\sqrt{n})$ approximation algorithm slightly improving on the result of Nagarajan and Sviridenko [22]. The new algorithm is surprisingly simple. It is based on a rounding of a natural LP relaxation. The LP relaxation is due to Adams and Johnson [1]. Thus we show that the integrality gap of the LP is $O(\sqrt{n})$.

Consider the following integer program. We have assignment variables x_{up} between vertices of the two graphs that are indicator variables of the events "*u maps to p*", and variables y_{upvq} that are indicator variables of the events "*u maps to p and v maps to q*". The LP relaxation is obtained by dropping the integrality condition on variables.

LP Relaxation

$$\max \sum_{\substack{u,v \in V_G \\ p,q \in V_H}} w_G(u,v) w_H(p,q) y_{upvq}$$

$$\begin{aligned}
&\textstyle\sum_{p \in V_H} x_{up} = 1, && \text{for all } u \in V_G;\\
&\textstyle\sum_{u \in V_G} x_{up} = 1, && \text{for all } p \in V_H;\\
&\textstyle\sum_{u \in V_G} y_{upvq} = x_{vq}, && \text{for all } u \in V_G,\ p,q \in V_H;\\
&\textstyle\sum_{p \in V_H} y_{upvq} = x_{vq}, && \text{for all } u,v \in V_G,\ q \in V_H;\\
&y_{upvq} = y_{vqup}, && \text{for all } u,v \in V_G,\ p,q \in V_H;\\
&x_{up} \in [0,1], && \text{for all } u \in V_G,\ p \in V_H;\\
&y_{upvq} \in [0,1], && \text{for all } u \in V_G,\ p \in V_H.
\end{aligned}$$

Approximation Algorithm

1. We solve the LP relaxation and obtain an optimal solution (x^*, y^*). Then we pick random subsets of vertices $L_G \subset V_G$ and $L_H \subset V_H$ of size $\lfloor n/2 \rfloor$. Let $R_G = V_G \setminus L_G$ and $R_H = V_H \setminus L_H$. In the rest of the algorithm, we will care only about edges going from L_G to R_G and from L_H to R_H; and we will ignore edges that completely lie in L_G, R_G, L_H or R_H.
2. For every vertex u in the set L_G, we pick a vertex p in L_H with probability x^*_{up} and set $\widetilde{\varphi}(u) = p$ (recall that $\sum_p x^*_{up} = 1$, for all u; with probability $1 - \sum_{p \in L_H} x^*_{up}$ we do not choose any vertex for u). Then for every vertex $p \in L_H$, which is chosen by at least one element u, we pick one of these u's

uniformly at random; and set $\varphi(u) = p$ (in other words, we choose a random $u \in \widetilde{\varphi}^{-1}(p)$ and set $\varphi(u) = p$). Let $\widetilde{L}_G \subset L_G$ be the set of all chosen u's.

3. We now find a permutation $\psi : R_G \to R_H$ so as to maximize the contribution we get from edges from $\widetilde{L}_G$ to R_G i.e., to maximize the sum

$$\sum_{\substack{u \in \widetilde{L}_G \\ v \in R_G}} w_G(u,v) w_H(\varphi(u), \psi(v)).$$

This can be done, since the problem is equivalent to the maximum matching problem between the sets R_G and R_H where the weight of the edge from v to q equals

$$\sum_{u \in \widetilde{L}_G} w_G(u,v) w_H(\varphi(u), q).$$

4. Output the mapping φ for vertices in $\widetilde{L}_G$, mapping ψ for vertices in R_G, and an arbitrary mapping for vertices in $L_G \setminus \widetilde{L}_G$, consistent with φ and ψ.

Remark 1. In the full version of the paper we also give a de-randomized version of the algorithm.

4.1 Analysis of the Algorithm

Theorem 3. *The approximation ratio of the algorithm is* $O(\sqrt{n})$.

While the algorithm is really simple, the analysis is more involved. Let LP^* be the value of the LP solution. To prove that the algorithm gives $O(\sqrt{n})$-approximation, it suffices to show that

$$\mathbb{E}\left[\sum_{\substack{u \in L_G \\ v \in R_G}} w_G(u,v) w_H(\varphi(u), \psi(v))\right] \geq \frac{LP^*}{O(\sqrt{n})}. \tag{2}$$

We split all edges of graph G into two sets: heavy edges and light edges. For each vertex $u \in V_G$, let $\mathcal{W}_u$ be the set of $\sqrt{n}$ vertices $v \in V_G$ with the largest weight $w_G(u,v)$. Then, $LP^* =$

$$\sum_{\substack{u \in V_G \\ v \in V_G \setminus \mathcal{W}_u}} \sum_{p,q \in V_H} y^*_{upvq} w_G(u,v) w_H(p,q) + \sum_{\substack{u \in V_G \\ v \in \mathcal{W}_u}} \sum_{p,q \in V_H} y^*_{upvq} w_G(u,v) w_H(p,q).$$

Denote the first term by LP^*_I and the second by LP^*_{II}. Instead of working with ψ, we explicitly define two new bijective maps ν_I and ν_{II} from R_G to R_H and prove, that $\mathbb{E}\left[\sum_{\substack{u \in \widetilde{L}_G \\ v \in R_G}} w_G(u,v) w_H(\varphi(u), \nu_I(v))\right] \geq \frac{LP^*_I}{O(\sqrt{n})}$ and

$\mathbb{E}\left[\sum_{\substack{u\in\widetilde{L}_G\\ v\in R_G}} w_G(u,v)w_H(\varphi(u),\nu_{II}(v))\right] \geq \frac{LP_{II}^*}{O(\sqrt{n})}$. These two inequalities imply the bound we need, since the sum (2) is greater than or equal to each of the sums above.

The first map ν_I is a random permutation between R_G and R_H. Observe, that given subsets L_G and L_H, the events $\{\widetilde{\varphi}(u) = p\}$ are mutually independent for different u's and the expected size of $\widetilde{\varphi}^{-1}(p)$ is at most 1, here $\widetilde{\varphi}^{-1}(p)$ is the preimage of p (recall the map $\widetilde{\varphi}$ may have collisions, and hence $\widetilde{\varphi}^{-1}(p)$ may contain more than one element). Thus (we give the details in the full version),

$$\Pr\{\varphi(u) = p \quad | \quad L_G, L_H\} \geq \begin{cases} x^*_{up}/2, & \text{if } u \in L_G \text{ and } p \in L_H; \\ 0, & \text{otherwise.} \end{cases}$$

For every $u, v \in V_G$ and $p, q \in V_H$, let $\mathcal{E}_{upvq}$ be the event $\{u \in L_G, v \in R_G, p \in L_H, q \in R_H\}$. Then, $\Pr\{\mathcal{E}_{upvq}\} = \Pr\{u \in L_G, v \in R_G, p \in L_H, q \in R_H\} = \frac{1}{16} - o(1)$. Thus, the probability that $\varphi(u) = p$ and $\nu_I(u) = q$ is $\Omega(x^*_{up}/n)$. We have

$$\begin{aligned}
&\mathbb{E} \sum_{\substack{u\in L_G\\ v\in R_G}} w_G(u,v)w_H(\varphi(u),\nu_I(v)) \\
&\qquad \geq \Omega(1) \times \sum_{u,v\in V_G} \sum_{p,q\in V_H} \frac{x^*_{up}}{n}\, w_G(u,v)w_H(p,q) \\
&\qquad \geq \Omega(1) \times \sum_{p,q\in V_H} w_H(p,q) \sum_{u\in V_G} x^*_{up} \sum_{v\in\mathcal{W}_u} \frac{w_G(u,v)}{n} \\
&\qquad \geq \Omega(1) \times \sum_{p,q\in V_H} w_H(p,q) \sum_{u\in V_G} x^*_{up} \frac{\min\{w_G(u,v) : v \in \mathcal{W}_u\}}{\sqrt{n}}.
\end{aligned}$$

On the other hand,

$$\begin{aligned}
LP_I^* &= \sum_{p,q\in V_H} w_H(p,q) \sum_{u\in V_G} x^*_{up} \left(\sum_{v\in V_G\setminus\mathcal{W}_u} \frac{y^*_{upvq}}{x^*_{up}} w_G(u,v) \right) \\
&\leq \sum_{p,q\in V_H} w_H(p,q) \sum_{u\in V_G} x^*_{up} \max\{w_G(u,v) : v \in V_G \setminus \mathcal{W}_u\} \\
&\leq \sum_{p,q\in V_H} w_H(p,q) \sum_{u\in V_G} x^*_{up} \min\{w_G(u,v) : v \in \mathcal{W}_u\}.
\end{aligned}$$

We now define ν_{II}. For every $v \in V_G$, let

$$l(v) = \text{argmax}_{u\in V_G} \left\{ \sum_{p,q\in V_H} w_G(u,v)w_H(p,q)y_{upvq} \right\}.$$

We say that $(l(v), v)$ is a heavy edge. For every $u \in L_G$, let $\mathcal{R}_u = \{v \in R_G : l(v) = u\}$. All sets $\mathcal{R}_u$ are disjoint subsets of R_G. We now define a map $\widetilde{\nu}_{II} : \mathcal{R}_u \to R_H$ independently for each $\mathcal{R}_u$ for which $\widetilde{\varphi}(u)$ is defined (even if $\varphi(u)$ is not defined). For every $v \in \mathcal{R}_u$, and $q \in R_H$, define $z_{vq} = \frac{y^*_{u\widetilde{\varphi}(u)vq}}{x^*_{u\widetilde{\varphi}(u)}}$. Observe, that $\sum_{v\in\mathcal{R}_u} z_{vq} \le 1$ for each $q \in R_H$ and $\sum_{q\in R_H} z_{vq} \le 1$ for each $v \in \mathcal{R}_u$. Hence, for a fixed $\mathcal{R}_u$, the vector $(z_{vq} : v \in \mathcal{R}_u, q \in R_H)$ lies in the convex hull of integral partial matchings between $\mathcal{R}_u$ and R_H. Thus, the fractional matching $(z_{vq} : v \in \mathcal{R}_u, q \in R_H)$ can be represented as a convex combination of integral partial matchings. Pick one of them with the probability proportional to its weight in the convex combination. Call this matching $\widetilde{\nu}^u_{II}$. Note, that $\widetilde{\nu}^u_{II}$ is injective and that the supports of $\widetilde{\nu}^{u'}_{II}$ and $\widetilde{\nu}^{u''}_{II}$ do not intersect if $u' \neq u''$ (since $\mathcal{R}_{u'} \cap \mathcal{R}_{u''} = \varnothing$). Let $\widetilde{\nu}_{II}$ be the union of $\widetilde{\nu}^u_{II}$ for all $u \in L_G$. The partial map $\widetilde{\nu}_{II}$ may not be injective and may map several vertices of R_G to the same vertex q. Thus, for every q in the image of R_G, we pick uniformly at random one preimage v and set $\nu_{II}(v) = q$. We define ν_{II} on the rest of R_G arbitrarily. In the full version of the paper, we show that

$$\mathbb{E} \sum_{\substack{u\in L_G\\ v\in R_G}} w_G(u,v) w_H(\varphi(u), \nu_{II}(v)) \ge \frac{1}{4}\,\mathbb{E}_{L_G,L_H}\left[\sum_{v\in R_G : l(v)\in L_G} \sum_{\substack{p\in L_H\\ q\in R_H}} y^*_{l(v)pvq}\, w_G(u,v) w_H(p,q)\right]$$

$$= \frac{1}{64+o(1)} \sum_{v\in V_G} \sum_{p,q\in V_H} y^*_{l(v)pvq}\, w_G(l(v),v) w_H(p,q)$$

$$= \frac{1}{64+o(1)} \sum_{v\in V_G} \max_{u\in V_G} \left\{ \sum_{p,q\in V_H} y^*_{upvq}\, w_G(u,v) w_H(p,q) \right\}$$

$$\ge \frac{1}{64+o(1)} \sum_{v\in V_G} \frac{1}{|\mathcal{W}_v|} \sum_{u\in\mathcal{W}_v} \sum_{p,q\in V_H} y^*_{upvq}\, w_G(u,v) w_H(p,q)$$

$$= \frac{1}{64+o(1)} \times \frac{LP^*_{II}}{\lceil\sqrt{n}\,\rceil}.$$

This finishes the proof.

Acknowledgement

We would like to thank anonymous referees for valuable comments and suggestions.

References

1. Adams, W.P., Johnson, T.A.: Improved Linear Programming-based Lower Bounds for the Quadratic Assignment Problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16, pp. 43–77 (1994)
2. Anstreicher, K.: Recent advances in the solution of quadratic assignment problems. In: Anstreicher, K. (ed.) ISMP, 2003. Copenhagen. Math. Program, Ser. B, vol. 97(1-2), pp. 27–42 (2003)
3. Arkin, E., Hassin, R., Sviridenko, M.: Approximating the Maximum Quadratic Assignment Problem. Information Processing Letters 77, 13–16 (2001)
4. Arora, S., Frieze, A., Kaplan, H.: A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. Mathematical Programming 92(1), 1–36 (2002)
5. Arora, S., Lund, C.: Hardness of Approximations. In: Hochbaum, D. (ed.) Approximation Algorithms for NP-hard Problems. PWS Publishing (1996)
6. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of the ACM 45(3)
7. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. Journal of the ACM 45(1), 70–122
8. Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting High Log-Densities – an $O(n^{1/4})$ Approximation for Densest k-Subgraph. In: Proceedings of STOC (to appear, 2010)
9. Burkard, R.E., Cela, E., Pardalos, P., Pitsoulis, L.S.: The quadratic assignment problem. In: Du, D.Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, vol. 3, pp. 241–339. Kluwer Academic Publishers, Dordrecht (1998)
10. Burkard, R.E., Dell'Amico, M., Martello, S.: Assignment Problems. SIAM, Philadelphia (2009)
11. Cela, E.: The Quadratic Assignment Problem: Theory and Algorithms. Springer, Heidelberg (1998)
12. Dong, Y., Wolkowicz, H.: A Low-Dimensional Semidefinite Relaxation for the Quadratic Assignment Problem. Mathematics of Operations Research 34, 1008–1022 (2009)
13. Dickey, J., Hopkins, J.: Campus building arrangement using TOPAZ. Transportation Science 6, 59–68 (1972)
14. Eiselt, H., Laporte, G.: A combinatorial optimization problem arising in dartboard design. Journal of Operational Research Society 42, 113–118 (1991)
15. Elshafei, A.: Hospital layout as a quadratic assignment problem. Operations Research Quarterly 28, 167–179 (1977)
16. Feige, U.: Relations between average case complexity and approximation complexity. In: Proceedings of STOC 2002 , pp. 534–543 (2002)
17. Frieze, A., Kannan, R.: Quick approximation to matrices and applications. Combinatorica 19(2), 175–220 (1999)
18. Geoffrion, A., Graves, G.: Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/LP approach. Operations Research 24, 596–610 (1976)
19. Hassin, R., Levin, A., Sviridenko, M.: Approximating the minimum quadratic assignment problems. ACM Transactions on Algorithms 6(1) (2009)
20. Khot, S.: Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In: Proceedings of FOCS 2004, pp. 136–145 (2004)

21. Koopmans, T.C., Beckman, M.: Assignment problems and the location of economic activities. Econometrica 25, 53–76 (1957)
22. Nagarajan, V., Sviridenko, M.: On the maximum quadratic assignment problem. Mathematics of Operations Research 34(4), 859–868 (2009); preliminary version appeared in Proceedings of SODA 2009, pp. 516–524
23. Laporte, G., Mercure, H.: Balancing hydraulic turbine runners: A quadratic assignment problem. European Journal of Operations Research 35, 378–381 (1988)
24. Loilola, E.M., De Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P.M., Querido, T.: A survey for the quadratic assignment problem. Invited Review, European Journal of Operational Research 176, 657–690 (2006)
25. Pardalos, P., Wolkowitz, H. (eds.): Proceedings of the DIMACS Workshop on Quadratic Assignment Problems. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16 (1994)
26. Papadimitriou, C.H., Yannakakis, M.: The traveling salesman problem with distances one and two. Mathematics of Operations Research 18, 1–11 (1993)
27. Raz, R.: A Parallel Repetition Theorem. SIAM Journal on Computing 27, 763–803 (1998)
28. Queyranne, M.: Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. Operations Research Letters 4, 231–234 (1986)
29. Steinberg, L.: The backboard wiring problem: a placement algorithm. SIAM Rev. 3, 37–50 (1961)
30. Zhao, Q., Karisch, S.E., Rendl, F., Wolkowicz, H.: Semidefinite Programming Relaxations for the Quadratic Assignment Problem. Journal of Combinatorial Optimization 2, 71–109 (1998)

Cell Probe Lower Bounds and Approximations for Range Mode

Mark Greve, Allan Grønlund Jørgensen,
Kasper Dalgaard Larsen, and Jakob Truelsen

MADALGO*, Department of Computer Science, Aarhus University, Denmark
{mgreve,jallan,larsen,jakobt}@madalgo.au.dk

Abstract. The mode of a multiset of labels, is a label that occurs at least as often as any other label. The input to the range mode problem is an array A of size n. A range query $[i, j]$ must return the mode of the subarray $A[i], A[i+1], \ldots, A[j]$. We prove that any data structure that uses S memory cells of w bits needs $\Omega(\frac{\log n}{\log(Sw/n)})$ time to answer a range mode query. Secondly, we consider the related range k-frequency problem. The input to this problem is an array A of size n, and a query $[i, j]$ must return whether there exists a label that occurs precisely k times in the subarray $A[i], A[i+1], \ldots, A[j]$. We show that for any constant $k > 1$, this problem is equivalent to 2D orthogonal rectangle stabbing, and that for $k = 1$ this is no harder than four-sided 3D orthogonal range emptiness. Finally, we consider approximate range mode queries. A c-approximate range mode query must return a label that occurs at least $1/c$ times that of the mode. We describe a linear space data structure that supports 3-approximate range mode queries in constant time, and a data structure that uses $O(\frac{n}{\varepsilon})$ space and supports $(1+\varepsilon)$-approximation queries in $O(\log \frac{1}{\varepsilon})$ time.

1 Introduction

In this paper we consider the range mode problem, the range k-frequency problem, and the c-approximate range mode problem. The frequency of a label l in a multiset S of labels, is the number of occurrences of l in S. The mode of S is the most frequent label in S. In case of ties, any of the most frequent labels in S can be designated the mode.

For all the problems we consider the input is an array A of length n containing labels. For simplicity we assume that each label is an integer between one and n. In the range mode problem, we must preprocess A into a data structure that given indices i and j, $1 \leq i \leq j \leq n$, returns the mode, $M_{i,j}$, in the subarray $A[i, j] = A[i], A[i+1], \ldots, A[j]$. We let $F_{i,j}$ denote the frequency of $M_{i,j}$ in $A[i, j]$. In the c-approximate range mode problem, a query is given indices i and j, $1 \leq i \leq j \leq n$, and returns a label that has a frequency of at least $F_{i,j}/c$.

* Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 605–616, 2010.

In the range k-frequency problem, a query is given indices i and j, $1 \leq i \leq j \leq n$, and returns whether there is a label occurring exactly k times in $A[i, j]$.

For the upper bounds we consider the unit cost RAM with word size $w = \Theta(\log n)$. For lower bounds we consider the cell probe model of Yao [1]. In this model of computation a random access memory is divided into cells of w bits. The complexity of an algorithm is the number of memory cells the algorithm accesses. All other computations are free.

Previous Results. The first data structure supporting range mode queries in constant time was developed in [2], and this data structure uses $O(n^2 \log\log n/\log n)$ space. This was subsequently improved to $O(n^2/\log n)$ space in [3] and finally to $O(n^2 \log\log n/\log^2 n)$ in [4]. For non-constant query time, the first data structure developed uses $O(n^{2-2\varepsilon})$ space and answers queries in $O(n^\varepsilon \log n)$ time, where $0 < \varepsilon \leq \frac{1}{2}$ is a query-space tradeoff constant [2]. The query time was later improved to $O(n^\varepsilon)$ without changing the space bound [3].

Given the rather large bounds for the range mode problem, the approximate variant of the problem was considered in [5]. With constant query time, they solve 2-approximate range mode with $O(n \log n)$ space, 3-approximate range mode with $O(n \log\log n)$ space, and 4-approximate range mode with linear space. For $(1+\varepsilon)$-approximate range mode, they describe a data structure that uses $O(\frac{n}{\varepsilon})$ space and answers queries in $O(\log\log_{(1+\varepsilon)} n) = O(\log\log n + \log\frac{1}{\varepsilon})$ time. This data structure gives a linear space solution with $O(\log\log n)$ query time for c-approximate range mode when c is constant. There are no known non-trivial lower bounds for the any of the problems we consider.

Our Results. In this paper we show the first lower bounds for range mode data structures and range k-frequency data structures and provide new upper bounds for the c-approximate range mode problem and the range k-frequency problem.

In Section 2 we prove our lower bound for range mode data structures. Specifically, we prove that any data structure that uses S cells and supports range mode queries must have a query time of $\Omega(\frac{\log n}{\log(Sw/n)})$. This means that any data structure that uses $O(n \log^{O(1)} n)$ space needs $\Omega(\log n/\log\log n)$ time to answer a range mode query. Similarly, any data structure that supports range mode queries in constant time needs $n^{1+\Omega(1)}$ space.

We suspect that the actual lower bound for near-linear space data structures for the range mode problem is significantly larger. However a fundamental obstacle in the cell probe model is to prove lower bounds for static data structures that are higher than the number of bits needed to describe the query. The highest known lower bounds are achieved by the techniques in [6,7] that uses reductions from problems in communication complexity. We use this technique to obtain our lower bound and our bound matches the highest lower bound achieved with this technique.

Actually our construction proves the same lower bound for queries on the form, is there an element with frequency at least (or precisely) k in $A[i, j]$, where k is given at query time. In the scenario where k is fixed for all queries it is trivial to give a linear space data structure with constant query time for determining

whether there is an element with frequency at least k. In Section 3 we consider the case of determining whether there is an element with frequency exactly k, which we denote the range k-frequency problem. To the best of our knowledge, we are the first to consider this problem. We show that 2D rectangle stabbing reduces to range k-frequency for any constant $k > 1$. This reduction proves that any data structure that uses S space, needs $\Omega(\log n/\log(Sw/n))$ time for a query [7,8], for any constant $k > 1$. Secondly, we reduce range k-frequency to 2D rectangle stabbing. This reduction works for any k. This immediately gives a data structure for range k-frequency that uses linear space, and answers queries in optimal $O(\log n/\log\log n)$ time [9] (we note that 2D rectangle stabbing reduces to 2D range counting). In the restricted case where $k = 1$, this problem corresponds to determining whether there is a unique label in a subarray. The reduction from 2D rectangle stabbing only applies for $k > 1$. We show, somewhat surprisingly, that determining whether there is a label occurring exactly twice (or $k > 1$ times) in a subarray, is exponentially harder than determining if there is a label occurring exactly once. Specifically, we reduce range 1-frequency to four-sided 3D orthogonal range emptiness, which can be solved with $O(\log^2 \log n)$ query time and $O(n \log n)$ space by a slight modification of the data structure presented in [10].

In Section 4 we present a simple data structure for the 3-approximate range mode problem. The data structure uses linear space and answers queries in constant time. This improves the best previous 3-approximate range mode data structures by a factor $O(\log\log n)$ either in space or query time. With linear space and constant query time, the best previous approximation factor was 4. In Section 5 we use our 3-approximate range mode data structure, to develop a data structure for $(1+\varepsilon)$-approximate range mode. This data structure uses $O(\frac{n}{\varepsilon})$ space and answers queries in $O(\log \frac{1}{\varepsilon})$ time. This removes the dependency on n in the query time compared to the previously best data structure, while matching the space bound. Thus, we have a linear space data structure with constant query time for the c-approximate range mode problem for any constant $c > 1$. We note that we get the same bound if we build on the 4-approximate range mode data structure from [5].

2 Cell Probe Lower Bound for Range Mode

In this section we show a query lower bound of $\Omega(\log n/\log(Sw/n))$ for any range mode data structure that uses S space for an input array of size n. The lower bound is proved for the slightly different problem of determining the frequency of the mode. Since the frequency of an element in any range can be determined in $O(\log\log n)$ time by a linear space data structure the lower bound for range mode follows. This data structure stores a linear space static rank data structure [11] for each label ℓ in the input, containing the positions in A storing ℓ. The frequency of a label in $A[i,j]$ is the rank difference between $i-1$ and j.

Communication Complexity and Lower Bounds. In communication complexity we have two players Alice and Bob. Alice receives as input a bit string x and

Bob a bit string y. Given some predefined function, f, the goal for Alice and Bob is to compute $f(x, y)$ while communicating as few bits as possible.

Lower bounds on the communication complexity of various functions have been turned into lower bounds for static data structure problems in the cell probe model. The idea is as follows [12]: Assume we are given a static data structure problem and consider the function $f(q, D)$ that is defined as the answer to a query q on an input set D for this problem. If we have a data structure for the problem that uses S memory cells and supports queries in time t we get a communication protocol for f where Alice sends $t \log S$ bits and Bob sends tw bits. In this protocol Alice receives q and Bob receives D. Bob constructs the data structure on D and Alice simulates the query algorithm. In each step Alice sends $\log S$ bits specifying the memory cell of the data structure she needs and Bob replies with the w bits of this cell. Finally, Alice outputs $f(q, D)$. Thus, a communication lower bound for f gives a lower bound tradeoff between S and t.

This construction can only be used to distinguish between polynomial and superpolynomial space data structures. Since range mode queries are trivially solvable in constant time with $O(n^2)$ space, we need a different technique to obtain lower bounds for near-linear space data structures. Pătraşcu and Thorup [6,7] have developed a technique for distinguishing between near linear and polynomial space by considering reductions from communication complexity problems to k parallel data structure queries. The main insight is that Alice can simulate all k queries in parallel and only send $\log \binom{S}{k} = O(k \log \frac{S}{k})$ bits to define the k cells she needs. For the right values of k this is significantly less than $k \log S$ bits which Alice needs if she performs the queries sequentially.

Lopsided Set Disjointness (LSD). In LSD Alice and Bob receive subsets S and T of a universe U. The goal for Alice and Bob is to compute whether $S \cap T \neq \emptyset$. LSD is parameterized with the size $|S| = N$ of Alice's set and the fraction between the size of the universe and N, which is denoted B, e.g. $|U| = NB$. Notice that the size of Bob's set is arbitrary and could be as large as NB. We use $[X]$ to denote the set $\{1, 2, \ldots, X\}$. There are other versions of LSD where the input to Alice has more structure. For our purpose we need Blocked-LSD. For this problem the universe is considered as the cartesian product of $[N]$ and $[B]$, e.g. $U = [N] \times [B]$ and Alice receives a set S such that $\forall j \in [N]$ there exists a unique $b_j \in [B]$ such that $(j, b_j) \in S$, e.g. S is of the form $\{(1, b_1), (2, b_2), \ldots, (N, b_N) \mid b_i \in [B]\}$. The following lower bound applies for this problem [7].

Theorem 1. *Fix $\delta > 0$. In a bounded-error protocol for Blocked-LSD, either Alice sends $\Omega(N \log B)$ bits or Bob sends $\Omega(NB^{1-\delta})$ bits.*

Blocked-LSD reduces to N/k parallel range mode queries. Given n, we describe a reduction from Blocked-LSD with a universe of size n ($n = NB$) to N/k parallel range mode queries on an input array A of size $\Theta(n)$. The size of A may not be exactly n but this will not affect our result. The parameters k and B are fixed later in the construction. From a high level perspective we construct an array of permutations of $[kB]$. A query consists of a suffix of one permutation, a number of complete permutations, and a prefix of another permutation. They are chosen

such that the suffix determines a subset of Bob's set and the prefix a subset of Alice's set. These two subsets intersect if and only if the frequency of the mode is equal to two plus the number of complete permutations spanned by the query.

Bob stores a range mode data structure and Alice simulates the query algorithm. First we describe the array A that Bob constructs when he receives his input. Let $T \subseteq [N] \times [B]$ be this set. The array Bob constructs consists of two parts which are described separately. We let $\cdot$ denote concatenation of lists. We also use this operator on sets and in this case we treat the set as a list by placing the elements in lexicographic order. Bob partitions $[N]$ into N/k consecutive chunks of k elements, e.g. the i'th chunk is $\{(i-1)k+1,\ldots,ik\}$ for $i = 1,\ldots,N/k$. With the i'th chunk Bob associates the subset L_i of T with first coordinate in that chunk, e.g. $L_i = T \cap (\{(i-1)k+t \mid t = 1,\ldots,k\} \times [B])$. Each L_i is mapped to a permutation of $[kB]$.

We define the mapping $f : (x, y) \to (x - 1 \bmod k)B + y$ and let the permutation be $([kB] \setminus f(L_i)) \cdot f(L_i)$, e.g. we map the elements in L_i into $[kB]$ and prepend the elements of $[kB]$ not mapped to by any element in L_i such that we get a full permutation of $[kB]$. The first part of A is the concatenation of the permutations defined for each chunk L_i ordered by i, e.g. $([kB] \setminus f(L_1)) \cdot f(L_1) \cdots ([kB] \setminus f(L_{N/k})) \cdot f(L_{N/k})$. The second part of A consists of B^k permutations of $[kB]$. There is one permutation for each way of picking a set of the form $\{(1, b_1),\ldots,(k, b_k) \mid b_i \in [B]\}$. Let $R_1,\ldots,R_{B^k}$ denote the B^k sets on this form ordered lexicographically. The second part of the array becomes $f(R_1) \cdot ([kB] \setminus f(R_1)) \cdots f(R_{B^k}) \cdot ([kB] \setminus f(R_{B^k}))$.

We now show how Alice and Bob can determine whether $S \cap T \neq \emptyset$ from this array. Bob constructs a range mode data structure for A and sends $|L_i|$ for $i = 1,\ldots,N/k$ to Alice. Alice then simulates the query algorithm on the range mode data structure for N/k queries in parallel. The i'th query determines whether the k elements $Q_i = \{((i-1)k+1, b_{(i-1)k+1}),\ldots,(ik, b_{ik})\}$ from S have an empty intersection with T (actually L_i) as follows.

Alice determines the end index of $f(Q_i)$ in the second part of A. We note that $f(Q_i)$ always exists in the second part of A by construction and Alice can determine the position without any communication with Bob. Alice also determines the start index of $f(L_i)$ in the first part of A from the sizes she initially received from Bob. The i'th query computes the frequency R_i of the mode between these two indices. Let p be the number of permutations of $[kB]$ stored between the end of $f(L_i)$ and the beginning of $f(Q_i)$ in A, then $F_i - p = 2$ if and only if $Q_i \cap T \neq \emptyset$, and $F_i - p = 1$ otherwise. Since each permutation of $[kB]$ contributes one to F_i, $F_i - p$ is equal to two if and only if at least one of the elements from Q_i is in L_i meaning that $S \cap T \neq \emptyset$. We conclude that Blocked-LSD reduces to N/k range mode queries in an array of size $NB + B^k kB$.

To obtain a lower bound for range mode data structures we consider the parameters k and B and follow the approach from [7]. Let S be the size of Bob's range mode data structure and let t be the query time. In our protocol for Blocked-LSD Alice sends $t \log \binom{S}{N/k} = O(t\frac{N}{k} \log \frac{Sk}{N})$ bits and Bob sends $twN/k + N/k \log(kB)$ bits. By Theorem 1, either Alice sends $\Omega(N \log B)$ bits

or Bob sends $\Omega(NB^{1-\delta})$. Fix $\delta = \frac{1}{2}$. Since $N/k \log(kB) = o(N\sqrt{B})$ we obtain that either $t\frac{N}{k}\log(\frac{Sk}{N}) = \Omega(N \log B)$ or $twN/k = \Omega(N\sqrt{B})$. We constrain B such that $B \geq w^2$ and $\log B \geq \frac{1}{2}\log(\frac{Sk}{N}) \Rightarrow B \geq \frac{Sk}{n}$ and obtain $t = \Omega(k)$. Since $|A| = NB + B^k kB$ and we require $|A| = \Theta(n)$, we set $k = \Theta(\log_B n)$. To maximize k we choose $B = \max\{w^2, \frac{Sk}{n}\}$. We obtain that $t = \Omega(k) = \Omega(\log N/\log\frac{Swk}{n}) = \Omega(\log n/\log\frac{Sw}{n})$ since $w > k$.

Summarizing, we get the following theorem.

Theorem 2. *Any data structure that uses S space needs $\Omega\left(\frac{\log n}{\log(\frac{Sw}{n})}\right)$ time for a range mode query in an array of size n.*

It follows from the construction that we get the same lower bound for data structures that support queries that are given i, j and k, and returns whether there exists an element with frequency exactly k in $A[i, j]$ or support queries that are given i, j and k and returns whether there is an element with frequency at least k in $A[i, j]$.

3 Range k-Frequency

In this section, we consider the *range k-frequency* problem and its connection to classic geometric data structure problems. We show that the range k-frequency problem is equivalent to 2D rectangle stabbing for any fixed constant $k > 1$, and that for $k = 1$ the problem reduces to four-sided 3D orthogonal range emptiness.

In the 2D rectangle stabbing problem the input is n axis-parallel rectangles. A query is given a point, (x, y), and must return whether this point is contained[1] in at least one of the n rectangles in the input. A query lower bound of $\Omega(\log n/\log(Sw/n))$ for data structures using S space is proved in [7], and a linear space static data structure with optimal $O(\log n/\log\log n)$ query time can be found in [9].

In four-sided 3D orthogonal range emptiness, we are given a set P of n points in 3D, and must preprocess P into a data structure, such that given an open-ended four-sided rectangle $R = (-\infty, x] \times [y_1, y_2] \times [z, \infty)$, the data structure returns whether R contains a point $p \in P$. Currently, the best solution for this problem uses $O(n \log n)$ space and supports queries in $O(\log^2 \log n)$ time [10].

For simplicity, we assume that each coordinate is a unique integer between one and $2n$ (rank space).

Theorem 3. *Let k be a constant greater than one. The 2D rectangle stabbing problem reduces to the range k-frequency problem.*

Proof. We show the reduction for $k = 2$ and then generalize this construction to any constant value $k > 2$.

Let $R_1, \ldots, R_n$ be the input to the rectangle stabbing problem. We construct a range 2-frequency instance with n distinct labels each of which is duplicated

[1] Points on the border of a rectangle are contained in the rectangle.

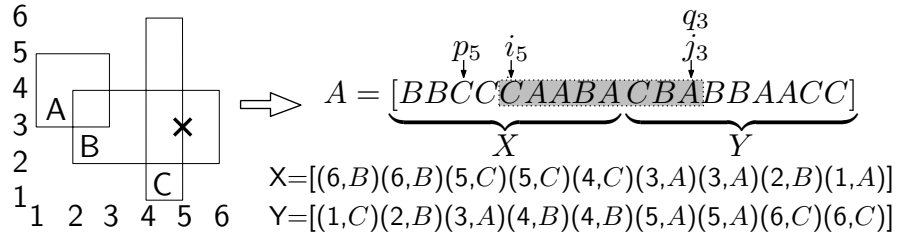


Fig. 1. Reduction from 2D rectangle stabbing to range 2-frequency. The × marks a stabbing query, $(5, 3)$. This query is mapped to the range 2-frequency query $[i_5, |X|+j_3]$ in A, which is highlighted. Notice that $i_5 = p_5 + 2$ since $A[p_5] = A[p_5 + 1]$.

exactly 6 times. Let R_ℓ be the rectangle $[x_{\ell_0}, x_{\ell_1}] \times [y_{\ell_0}, y_{\ell_1}]$. For each rectangle, R_ℓ, we add the pairs (x_{ℓ_0}, ℓ), (x_{ℓ_1}, ℓ) and (x_{ℓ_1}, ℓ) to a list X. Similarly, we add the pairs (y_{ℓ_0}, ℓ), (y_{ℓ_1}, ℓ), and (y_{ℓ_1}, ℓ) to a list Y. We sort X in descending order and Y in ascending order by their first coordinates. Since we assumed all coordinates are unique, the only ties are amongst pairs originating from the same rectangle, here we break the ties arbitrarily. The concatenation of X and Y is the range 2-frequency instance and we denote it A, i.e. the second component of each pair are the actual entries in A, and the first component of each pair is ignored.

We translate a 2D rectangle stabbing query, (x, y), into a query for the range 2-frequency instance as follows. Let p_x be the smallest index where the first coordinate of $X[p_x]$ is x, and let q_y be the largest index where the first coordinate of $Y[p_y]$ is y. If $A[p_x] = A[p_x + 1]$, two consecutive entries in A are defined by the right endpoint of the same rectangle, we set $i_x = p_x + 2$ (we move i_x to the right of the two entries), otherwise we set $i_x = p_x$. Similarly for the y coordinates, if $A[|X| + q_y] = A[|X| + q_y - 1]$ we set $j_y = q_y - 2$ (move j_y left of the two entries), otherwise we set $j_y = q_y$. Finally we translate (x, y) to the range 2-frequency query $[i_x, |X|+j_y]$ on A, see Figure 1. Notice that in the range 2-frequency queries that can be considered in the reduction, the frequency of a label is either one, two, three, four or six. The frequency of label ℓ in $A[i_x, |X|]$ is one if $x_{\ell_0} \leq x \leq x_{\ell_1}$, three if $x > x_{\ell_1}$ and zero otherwise. Similar, the frequency of ℓ in $A[|X| + 1, |X| + j_y]$ is one if $y_{\ell_0} \leq y \leq y_{\ell_1}$, three if $y > y_{\ell_1}$ and zero otherwise. We conclude that the point (x, y) stabs rectangle R_ℓ if and only if the label ℓ has frequency two in $A[i_x, |X| + j_y]$.

Since $x, y \in \{1, \ldots, 2n\}$, we can store a table with the translations from x to i_x and y to j_y. Thus, we can translate 2D rectangle stabbing queries to range 2-frequency queries in constant time.

For $k > 2$ we place $k - 2$ copies of each label between X and Y and translate the queries accordingly. □

The following theorem provides a matching upper bound.

Theorem 4. *The range k-frequency problem reduces to 2D rectangle stabbing.*

Proof. Let A be the input to the range k-frequency problem. We translate the ranges of A where there is a label with frequency k into $O(n)$ rectangles as follows. Fix a label $x \in A$, and let $s_x \geq k$ denote the number of occurrences of x in A. If $s_x < k$ then x is irrelevant and we discard it. Otherwise, let $i_1 < i_2 < \ldots < i_s$ be the position of x in A, and let $i_0 = 0$ and $i_{s+1} = n + 1$. Consider the ranges of A where x has frequency k. These are the subarrays, $A[a, b]$, where there exists an integer ℓ such that $i_\ell < a \leq i_{\ell+1}$ and $i_{\ell+k} \leq b < i_{\ell+k+1}$ for $0 \leq \ell \leq s_x - k$. This defines $s_x - k + 1$ two dimensional rectangles, $[i_\ell + 1, i_{\ell+1}] \times [i_{\ell+k}, i_{\ell+k+1} - 1]$ for $\ell = 0, \ldots, s_x - k$, such that x has frequency k in $A[i, j]$ if and only if the point (i, j) stabs one of the $s_x - k + 1$ rectangles defined by x. By translating the ranges of A where a label has frequency k into the corresponding rectangles for all distinct labels in A, we get a 2D rectangle stabbing instance with $O(n)$ rectangles. □

This means that we get a data structure for the range k-frequency problem that uses $O(n)$ space and supports queries in $O(\log n / \log \log n)$ time.

Theorem 5. *For $k = 1$, the range k-frequency problem reduces to four-sided orthogonal range emptiness queries in 3D.*

Proof. For each distinct label $x \in A$, we map the ranges of A where x has frequency one (it is unique in the range) to a 3D point. Let $i_1 < i_2 < \ldots < i_s$ be the positions of x in A, and let $i_0 = 0$ and $i_{s+1} = n+1$. The label x has frequency one in $A[a, b]$ if there exist an integer ℓ such that $i_{\ell-1} < a \leq i_\ell \leq b < i_{\ell+1}$. We define s points, $P_x = \{(i_{\ell-1} + 1, i_\ell, i_{\ell+1} - 1) \mid 1 \leq \ell \leq s\}$. The label x has frequency one in the range $A[a, b]$ if and only if the four-sided orthogonal range query $[-\infty, a] \times [a, b] \times [b, \infty]$ contains a point from P_x (we say that x is inside range $[x_1, x_2]$ if $x_1 \leq x \leq x_2$). Therefore, we let $P = \bigcup_{x \in A} P_x$ and get a four-sided 3D orthogonal range emptiness instance with $O(n)$ points. □

Thus, we get a data structure for the range 1-frequency problem that uses $O(n \log n)$ space and supports queries in $O(\log^2 \log n)$ time and we conclude that for data structures using $O(n \log^{O(1)} n)$ space, the range k-frequency problem is exponentially harder for $k > 1$ than for $k = 1$.

4 3-Approximate Range Mode

In this section, we construct a data structure that given a range $[i, j]$ computes a 3-approximation of $F_{i,j}$.

We use the following observation also observed in [5]. If we can cover $A[i, j]$ with three disjoint subintervals $A[i, x], A[x + 1, y]$ and $A[y + 1, j]$ then we have $\frac{1}{3} F_{i,j} \leq \max\{F_{i,x}, F_{x+1,y}, F_{y+1,j}\} \leq F_{i,j}$.

First, we describe a data structure that uses $O(n \log \log n)$ space, and then we show how to reduce the space to $O(n)$. The data structure consists of a tree T of polynomial fanout where the i'th leaf stores $A[i]$, for $i = 1, \ldots, n$. For a node v let T_v denote the subtree rooted at v and let $|T_v|$ denote the number of leaves

in T_v. The fanout of node v is $f_v = \lceil\sqrt{|T_v|}\rceil$. The height of T is $\Theta(\log\log n)$. Along with T, we store a lowest common ancestor (LCA) data structure, which given indices i and j, finds the LCA of the leaves corresponding to i and j in T in constant time [13].

For every node $v \in T$, let $R_v = A[a, b]$ denote the consecutive range of entries stored in the leaves of T_v. The children $c_1, \ldots, c_{f_v}$ of v partition R_v into f_v disjoint subranges $R_{c_1} = A[a_{c_1}, b_{c_1}]$, $\ldots$, $R_{c_{f_v}} = A[a_{c_{f_v}}, b_{c_{f_v}}]$ each of size $O(\sqrt{|T_v|})$. For every pair of children c_r and c_s where $r < s-1$, we store $F_{a_{c_{r+1}}, b_{c_{s-1}}}$. Furthermore, for every child range R_{c_i} we store $F_{a_{c_i}, k}$ and $F_{k, b_{c_i}}$ for every prefix and suffix range of R_{c_i} respectively. To compute a 3-approximation of $F_{i,j}$, we find the LCA of i and j. This is the node v in T for which i and j lie in different child subtrees, say T_{c_x} and T_{c_y} with ranges $R_{c_x} = [a_{c_x}, b_{c_x}]$ and $R_{c_y} = [a_{c_y}, b_{c_y}]$. We then lookup the frequency $F_{a_{c_{x+1}}, b_{c_{y-1}}}$ stored for the pair of children c_x and c_y, as well as the suffix frequency $F_{i, b_{c_x}}$ stored for the range $A[i, b_{c_x}]$ and the prefix frequency $F_{a_{c_y}, j}$ stored for $A[a_{c_y}, j]$, and return the max of these.

Each node $v \in T$ uses $O(|T_v|)$ space for the frequencies stored for each of the $O(|T_v|)$ pairs of children, and for all the prefix and suffix range frequencies. Since each node v uses $O(|T_v|)$ space and the LCA data structure uses $O(n)$ space, our data structure uses $O(n \log\log n)$ space. A query makes one LCA query and computes the max of three numbers which takes constant time.

We just need one observation to bring the space down to $O(n)$. Consider a node $v \in T$. The largest possible frequency that can be stored for any pair of children of v, or for any prefix or suffix range of a child of v is $|T_v|$, and each such frequency can be represented by $b = 1 + \lfloor \log |T_v| \rfloor$ bits. We divide the frequencies stored in v into chunks of size $\lfloor \frac{\log n}{b} \rfloor$ and pack each of them in one word. This reduces the total space usage of the nodes on level i to $O(n/2^i)$. We conclude that the data structure uses $O(n)$ space and supports queries in constant time.

Theorem 6. *There exists a data structure for the* 3*-approximate range mode problem that uses* $O(n)$ *space and supports queries in constant time.*

5 $(1+\varepsilon)$-Approximate Range Mode

In this section, we describe a data structure using $O(\frac{n}{\varepsilon})$ space that given a range $[i, j]$, computes a $(1+\varepsilon)$-approximation of $F_{i,j}$ in $O(\log \frac{1}{\varepsilon})$ time. Our data structure consists of two parts. The first part solves all queries $[i, j]$ where $F_{i,j} \leq \lceil \frac{1}{\varepsilon} \rceil$ (small frequencies), and the latter solves the remaining. The first data structure also decides whether $F_{i,j} \leq \lceil \frac{1}{\varepsilon} \rceil$. We use that $\frac{1}{\log(1+\varepsilon)} = O(\frac{1}{\varepsilon})$ for any $0 < \varepsilon \leq 1$.

Small Frequencies. For $i = 1, \ldots, n$ we store a table, Q_i, of length $\lceil \frac{1}{\varepsilon} \rceil$, where the value in $Q_i[k]$ is the largest integer $j \geq i$ such that $F_{i,j} = k$. To answer a query $[i, j]$ we do a successor search for j in Q_i. If j does not have a successor in Q_i then $F_{i,j} > \lceil \frac{1}{\varepsilon} \rceil$, and we query the second data structure. Otherwise, let s be the index of the successor of j in Q_i, then $F_{i,j} = s$. The data structure uses $O(\frac{n}{\varepsilon})$ space and supports queries in $O(\log \frac{1}{\varepsilon})$ time.

Large Frequencies. For every index $1 \leq i \leq n$, define a list T_i of length $t = \lceil \log_{1+\varepsilon}(\varepsilon n) \rceil$, with the following invariant: For all j, if $T_i[k-1] < j \leq T_i[k]$ then $\lceil \frac{1}{\varepsilon}(1+\varepsilon)^k \rceil$ is a $(1+\varepsilon)$-approximation of $F_{i,j}$. The following assignment of values to the lists T_i satisfies this invariant:

Let $m(i,k)$ be the largest integer $j \geq i$ such that $F_{i,j} \leq \lceil \frac{1}{\varepsilon}(1+\varepsilon)^{k+1} \rceil - 1$. For T_1 we set $T_1[k] = m(1,k)$ for all $k = 1, \ldots, t$. For the remaining T_i we set

$$T_i[k] = \begin{cases} T_{i-1}[k] \text{ if } F_{i,T_{i-1}[k]} \geq \lceil \frac{1}{\varepsilon}(1+\varepsilon)^k \rceil + 1 \\ m(i,k) \text{ otherwise} \end{cases}$$

The n lists are sorted by construction. For T_1, it is true since $m(i,k)$ is increasing in k. For T_i, it follows that $F_{i,T_i[k]} \leq \lceil \frac{1}{\varepsilon}(1+\varepsilon)^{k+1} \rceil - 1 < F_{i,T_i[k+1]}$, and thus $T_i[k] < T_i[k+1]$ for any k.

Let s be the index of the successor of j in T_i. We know that $F_{i,T_i[s]} \leq \lceil \frac{1}{\varepsilon}(1+\varepsilon)^{s+1} \rceil - 1$, $F_{i,T_i[s-1]} \geq \lceil \frac{1}{\varepsilon}(1+\varepsilon)^{s-1} \rceil + 1$ and $T_i[s-1] < j \leq T_i[s]$. It follows that

$$\lceil \tfrac{1}{\varepsilon}(1+\varepsilon)^{s-1} \rceil + 1 \leq F_{i,j} \leq \lceil \tfrac{1}{\varepsilon}(1+\varepsilon)^{s+1} \rceil - 1 \tag{1}$$

and that $\lceil \frac{1}{\varepsilon}(1+\varepsilon)^s \rceil$ is a $(1+\varepsilon)$-approximation of $F_{i,j}$.

The second important property of the n lists, is that they only store $O(\frac{n}{\varepsilon})$ different indices, which allows for a space-efficient representation. If $T_{i-1}[k] \neq T_i[k]$ then the following $\lceil \frac{1}{\varepsilon}(1+\varepsilon)^{k+1} \rceil - 1 - \lceil \frac{1}{\varepsilon}(1+\varepsilon)^k \rceil - 1 \geq \lfloor (1+\varepsilon)^k \rfloor - 3$ entries, $T_{i+a}[k]$ for $a = 1, \ldots, \lfloor (1+\varepsilon)^k \rfloor - 3$, are not changed, hence we store the same index at least $\max\{1, \lfloor (1+\varepsilon)^k \rfloor - 2\}$ times. Therefore, the number of changes to the n lists, starting with T_1, is bounded by $\sum_{k=1}^{t} \frac{n}{\max\{1, \lfloor (1+\varepsilon)^k \rfloor - 2\}} = O(\frac{n}{\varepsilon})$. This was observed in [5], where similar lists are maintained in a partially persistent search tree [14].

We maintain these lists without persistence such that we can access any entry in any list T_i in constant time. Let $I = \{1, 1+t, \ldots, 1+\lfloor (n-1)/t \rfloor t\}$. For every $\ell \in I$ we store T_ℓ explicitly as an array S_ℓ. Secondly, for $\ell \in I$ and $k = 1, \ldots, \lceil \log_{1+\varepsilon} t \rceil$ we define a bit vector $B_{\ell,k}$ of length t and a change list $C_{\ell,k}$, where

$$B_{\ell,k}[a] = \begin{cases} 0 \text{ if } T_{\ell+a-1}[k] = T_{\ell+a}[k] \\ 1 \text{ otherwise} \end{cases}$$

Given a bit vector L, define $\mathrm{sel}(L,b)$ as the index of the b'th one in L. We set

$$C_{\ell,k}[a] = T_{\ell+\mathrm{sel}(B_{\ell,k},a)}[k] \; .$$

Finally, for every $\ell \in I$ and for $k = 1 + \lceil \log_{1+\varepsilon} t \rceil, \ldots, t$ we store $D_\ell[k]$ which is the smallest integer $z > \ell$ such that $T_z[k] \neq T_\ell[k]$. We also store $E_\ell[k] = T_{D_\ell[k]}[k]$. We store each bit vector in a rank and select data structure [15] that uses $O(\frac{n}{w})$ space for a bit vector of length n, and supports $rank(i)$ in constant time. A $\mathrm{rank}(i)$ query returns the number of ones in the first i bits of the input.

Each change list, $C_{l,k}$ and every D_ℓ and E_ℓ list is stored as an array. The bit vectors indicate at which indices the contents of the first $\lceil \log_{1+\varepsilon} t \rceil$ entries of

$T_\ell, \ldots, T_{\ell+t-1}$ change, and the change lists store what the entries change to. The D_ℓ and E_ℓ arrays do the same thing for the last $t - \lceil \log_{1+\varepsilon} t \rceil$ entries, exploiting that these entries change at most once in an interval of length t.

Observe that the arrays, $C_{\ell,k}, D_\ell[k]$ and $E_\ell[k]$, and the bit vectors, $B_{\ell,k}$ allow us to retrieve the contents of any entry, $T_i[k]$ for any i, k, in constant time as follows. Let $\ell = \lfloor i/t \rfloor t$. If $k > \lceil \log_{1+\varepsilon} t \rceil$ we check if $D_\ell[k] \leq i$, and if so we return $E_\ell[k]$, otherwise we return $S_\ell[k]$. If $k \leq \lceil \log_{1+\varepsilon} t \rceil$, we determine $r = \text{rank}(i - \ell)$ in $B_{\ell,k}$ using the rank and select data structure. We then return $C_{\ell,k}[r]$ unless $r = 0$ in which case we return $S_\ell[k]$.

We argue that this correctly returns $T_i[k]$. In the case where $k > \lceil \log_{1+\varepsilon} t \rceil$, comparing $D_\ell[k]$ to i indicates whether $T_i[k]$ is different from $T_\ell[k]$. Since $T_z[k]$ for $z = \ell, \ldots, i$ can only change once, $T_i[k] = E_\ell[k]$ in this case. Otherwise, $S_\ell[k] = T_\ell[k] = T_i[k]$. If $k \leq \lceil \log_{1+\varepsilon} t \rceil$, the rank r of $i - \ell$ in $B_{\ell,k}$, is the number of changes that has occurred in the k'th entry from list T_ℓ to T_i. Since $C_{\ell,k}[r]$ stores the value of the k'th entry after the r'th change, $C_{\ell,k}[r] = T_i[k]$, unless $r = 0$ in which case $T_i[k] = S_\ell[k]$.

The space used by the data structure is $O(\frac{n}{\varepsilon})$. We store $3\lceil \frac{n}{t} \rceil$ arrays, S_ℓ, D_ℓ and E_ℓ for $\ell \in I$, each using t space, in total $O(n)$. The total size of the change lists, $C_{\ell,k}$, is bounded by the number of changes across the T_i lists, which is $O(\frac{n}{\varepsilon})$ by the arguments above. Finally, the rank and select data structures, $B_{\ell,k}$, each occupy $O(\frac{t}{w}) = O(\frac{t}{\log n})$ words, and we store a total of $\lceil \frac{n}{t} \rceil \lceil \log_{1+\varepsilon} t \rceil$ such structures, thus the total space used by these is bounded by $O\left(\frac{t}{\log n} \frac{n}{t} \log_{1+\varepsilon} t\right) = O\left(\frac{n}{\varepsilon} \frac{\log(n \log(\varepsilon n))}{\log n}\right) = O\left(\frac{n}{\varepsilon}\right)$. We use that if $\lceil \frac{1}{\varepsilon} \rceil \geq n$ then we only store the small frequency data structure. We conclude that our data structures uses $O\left(\frac{n}{\varepsilon}\right)$ space.

To answer a query $[i, j]$, we first compute a 3-approximation of $F_{i,j}$ in constant time using the data structure from Section 4. Thus, we find $f_{i,j}$ satisfying $f_{i,j} \leq F_{i,j} \leq 3f_{i,j}$. Choose k such that $\lceil \frac{1}{\varepsilon}(1+\varepsilon)^k \rceil + 1 \leq f_{i,j} \leq \lceil \frac{1}{\varepsilon}(1+\varepsilon)^{k+1} \rceil - 1$ then the successor of j in T_i must be in one of the entries, $T_i[k], \ldots, T_i[k + O(\log_{1+\varepsilon} 3)]$. As stated earlier, the values of T_i are sorted in increasing order, and we find the successor of j using a binary search on an interval of length $O(\log_{1+\varepsilon} 3)$. Since each access to T_i takes constant time, we use $O(\log \log_{1+\varepsilon} 3) = O(\log \frac{1}{\varepsilon})$ time.

Theorem 7. *There exists a data structure for $(1+\varepsilon)$-approximate range mode that uses $O(\frac{n}{\varepsilon})$ space and supports queries in $O(\log \frac{1}{\varepsilon})$ time.*

The careful reader may have noticed that our data structure returns a frequency, and not a label that occurs approximately $F_{i,j}$ times. We can augment our data structure to return a label instead as follows.

We set $\varepsilon' = \sqrt{(1+\varepsilon)} - 1$, and construct our data structure from above. The *small frequency* data structure is augmented such that it stores the label $M_{i,Q_i[k]}$ along with $Q_i[k]$, and returns this in a query. The *large frequency* data structure is augmented such that for every update of $T_i[k]$ we store the label that caused the update. Formally, let $a > 0$ be the first index such that $T_{i+a}[k] \neq T_i[k]$. Next to $T_i[k]$ we store the label $L_i[k] = A[i + a - 1]$. In a query, $[i, j]$, let s be the index of the successor of j in T_i computed as above. If $s > 1$ we return the

label $L_i[s-1]$, and if $s = 1$ we return $M_{i,Q_i[\lceil 1/\varepsilon' \rceil]}$, which is stored in the small frequency data structure.

In the case where $s = 1$ we know that $\lceil \frac{1}{\varepsilon'} \rceil \leq F_{i,j} \leq \lceil \frac{1}{\varepsilon'}(1+\varepsilon')^2 \rceil - 1 = \lceil \frac{1}{\varepsilon'}(1+\varepsilon) \rceil - 1$ and we know that the frequency of $M_{i,Q_i[\lceil 1/\varepsilon' \rceil]}$ in $A[i,j]$ is at least $\lceil \frac{1}{\varepsilon'} \rceil$. We conclude that the frequency of $M_{i,Q_i[\lceil 1/\varepsilon' \rceil]}$ in $A[i,j]$ is a $(1+\varepsilon)$-approximation of $F_{i,j}$.

If $s > 1$ we know that $\lceil \frac{1}{\varepsilon'}(1+\varepsilon')^{s-1} \rceil + 1 \leq F_{i,j} \leq \lceil \frac{1}{\varepsilon'}(1+\varepsilon')^{s+1} \rceil - 1$ by equation (1), and that the frequency, f_L, of the label $L_i[s-1]$ in $A[i,j]$ is at least $\lceil \frac{1}{\varepsilon'}(1+\varepsilon')^{s-1} \rceil + 1$. This means that $F_{i,j} \leq \frac{1}{\varepsilon'}(1+\varepsilon')^{s+1} \leq (1+\varepsilon')^2 f_L = (1+\varepsilon) f_L$, and we conclude that f_L is a $(1+\varepsilon)$-approximation of $F_{i,j}$.

The space needed for this data structure is $O(\frac{n}{\varepsilon'}) = O(\frac{n(\sqrt{1+\varepsilon}+1)}{\varepsilon}) = O(\frac{n}{\varepsilon})$, and a query takes $O(\log \frac{1}{\varepsilon'}) = O(\log \frac{1}{\varepsilon} + \log(\sqrt{1+\varepsilon}+1)) = O(\log \frac{1}{\varepsilon})$ time.

References

1. Yao, A.C.C.: Should tables be sorted? J. ACM 28(3), 615–628 (1981)
2. Krizanc, D., Morin, P., Smid, M.H.M.: Range mode and range median queries on lists and trees. Nord. J. Comput. 12(1), 1–17 (2005)
3. Petersen, H.: Improved bounds for range mode and range median queries. In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008. LNCS, vol. 4910, pp. 418–423. Springer, Heidelberg (2008)
4. Petersen, H., Grabowski, S.: Range mode and range median queries in constant time and sub-quadratic space. Inf. Process. Lett. 109(4), 225–228 (2008)
5. Bose, P., Kranakis, E., Morin, P., Tang, Y.: Approximate range mode and range median queries. In: Proc. 22nd Symposium on Theoretical Aspects of Computer Science, pp. 377–388 (2005)
6. Patrascu, M., Thorup, M.: Higher lower bounds for near-neighbor and further rich problems. In: Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 646–654 (2006)
7. Pătraşcu, M.: (Data) structures. In: Proc. 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 434–443 (2008)
8. Pătraşcu, M.: Lower bounds for 2-dimensional range counting. In: Proc. 39th ACM Symposium on Theory of Computing, pp. 40–46 (2007)
9. JáJá, J., Mortensen, C.W., Shi, Q.: Space-efficient and fast algorithms for multidimensional dominance reporting and counting. In: Fleischer, R., Trippen, G. (eds.) ISAAC 2004. LNCS, vol. 3341, pp. 558–568. Springer, Heidelberg (2004)
10. Afshani, P.: On dominance reporting in 3D. In: Proc. of the 16th Annual European Symposium on Algorithms, pp. 41–51 (2008)
11. Willard, D.E.: Log-logarithmic worst-case range queries are possible in space theta(n). Inf. Process. Lett. 17(2), 81–84 (1983)
12. Miltersen, P.B., Nisan, N., Safra, S., Wigderson, A.: On data structures and asymmetric communication complexity. J. Comput. Syst. Sci. 57(1), 37–49 (1998)
13. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM J. Comput. 13(2), 338–355 (1984)
14. Driscoll, J.R., Sarnak, N., Sleator, D.D., Tarjan, R.E.: Making data structures persistent. Journal of Computer and System Sciences 38(1), 86–124 (1989)
15. Jacobson, G.J.: Succinct static data structures. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA (1988)

SDP Gaps for 2-to-1 and Other Label-Cover Variants

Venkatesan Guruswami[1,⋆], Subhash Khot[2,⋆⋆], Ryan O'Donnell[1,⋆⋆⋆], Preyas Popat[2,⋆⋆], Madhur Tulsiani[3,†], and Yi Wu[1,⋆⋆⋆]

[1] Computer Science Department
Carnegie Mellon University
[2] Computer Science Department
New York University
[3] School of Mathematics
Institute for Advanced Study

Abstract. In this paper we present semidefinite programming (SDP) gap instances for the following variants of the Label-Cover problem, closely related to the Unique Games Conjecture: (i) 2-to-1 Label-Cover; (ii) 2-to-2 Label-Cover; (iii) α-constraint Label-Cover. All of our gap instances have perfect SDP solutions. For alphabet size K, the integral optimal solutions have value: (i) $O(1/\sqrt{\log K})$; (ii) $O(1/\log K)$; (iii) $O(1/\sqrt{\log K})$.

Prior to this work, there were no known SDP gap instances for any of these problems with perfect SDP value and integral optimum tending to 0.

1 Introduction

1.1 The Unique Games Conjecture and Its Variants

Since its introduction in 2002, the *Unique Games Conjecture* (UGC) of Khot [8] has proved highly influential and powerful in the study of probabilistically checkable proofs (PCPs) and approximation algorithms. Assuming the UGC yields many strong — and often, optimal — hardness of approximation results that we have been unable to obtain assuming only P $\neq$ NP. Perhaps the acme of this line of research so far is the work of Raghavendra [12], who showed the following result:

Theorem 1. *([12], informally.) Let $\mathcal{C}$ be any bounded-arity constraint satisfaction problem (CSP). Assume the Unique Games Conjecture. Then for a certain*

⋆ Research supported by a Packard Fellowship and US-Israel BSF-2008293.
⋆⋆ Research supported by NSF CAREER grant CCF-0833228, NSF Expeditions grant CCF-0832795, and BSF grant 2008059.
⋆⋆⋆ Research supported by NSF grants CCF-0747250 and CCF-0915893, BSF grant 2008477, and Sloan and Okawa fellowships.
† Research supported by NSF Grant CCF-0832797.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 617–628, 2010.

semidefinite programming (SDP) relaxation of $\mathcal{C}$, the SDP gap for $\mathcal{C}$ is the same as the optimal polynomial-time approximability gap for $\mathcal{C}$, up to an additive constant $\epsilon > 0$ which can be arbitrarily small.

Unfortunately, because of the additive ϵ term, Raghavendra's work is not applicable (even granting the UGC or any related conjecture) for the important case of completely *satisfiable* CSPs; equivalently, PCPs with *perfect completeness*. A good example of this comes from *coloring problems*; e.g., the very well known problem of coloring 3-colorable graphs. The UGC does not help in deducing any hardness result for such problems. Indeed the first strong hardness result for it, due to Dinur, Mossel, and Regev [3], used instead certain variants of UGC which have perfect completeness, namely, the "2-to-1 Conjecture", the "2-to-2 Conjecture", and the "α-Constraint Conjecture". (These conjectures will be described formally in Section 3.) An instance of Label-Cover with α-constraints was also implicit in the result of Dinur and Safra [4] on the hardness of approximating minimum vertex cover.

Recently, several more works have needed to use these alternate conjectures with perfect completeness: e.g., O'Donnell and Wu [11] and Tang [16] on Max-3CSP, Guruswami and Sinop [6] on Max-k-Colorable-Subgraph.

1.2 Statements of the Conjectures

Let us briefly give some definitions so that we may state some of the aforementioned conjectures more precisely.

Definition 1. *A* Label-Cover *instance $\mathcal{L}$ is defined by a tuple $((V, E), R, \Psi)$. Here (V, E) is a graph, R is a positive integer and Ψ is a set of constraints (relations), one for each edge: $\Psi = \{\psi_e \subseteq \{1, \ldots, R\}^2 \mid e \in E\}$. A labeling A is a mapping $A : V \to [R]$. We say that an edge $e = (u, v)$ is* satisfied *by A if $(A(u), A(v)) \in \psi_e$. We define:*

$$\mathrm{OPT}(\mathcal{L}) = \max_{A:V\to[R]} \mathop{\mathbf{Pr}}_{e=(u,v)\in E}[(A(u), A(v)) \in \psi_e]$$

Here the probability is over the uniform distribution of edges, i.e. each edge is equally likely to be picked.

Definition 2. *A constraint $\psi \subseteq \{1, \ldots, R\}^2$ is said to be a d-to-1* projection *if there is a map $\pi : [R] \to [R]$ such that for each element $j \in [R]$ we have $|\pi^{-1}(j)| \leq d$, and $(i, j) \in \psi$ if and only if $j = \pi(i)$. A Label-Cover instance is said to be d-to-1 if all its constraints are d-to-1 projections.*

We now state some conjectures on the inapproximability of Unique and 2-to-1 Label-Cover. In Section 3.1 we will also discuss the two other variants of the Label-Cover problem, based on 2-to-2 and "α" constraints, and the associated 2-to-2 and α-Constraint Conjectures on their inapproximability.

Conjecture 1. [7] (**Unique Games Conjecture**) For any $\epsilon, \delta > 0$, it is NP-hard to decide whether a 1-to-1 bipartite Label-Cover instance $\mathcal{L}$ has $\mathrm{OPT}(\mathcal{L}) \geq 1 - \epsilon$ or has $\mathrm{OPT}(\mathcal{L}) \leq \delta$.

Notice that the above problem is in P when $\epsilon = 0$.

Conjecture 2. [7] (**2-to-1 Conjecture**) For any $\delta > 0$, it is NP-hard to decide whether a 2-to-1 bipartite Label-Cover instance $\mathcal{L}$ has $\mathrm{OPT}(\mathcal{L}) = 1$ or has $\mathrm{OPT}(\mathcal{L}) \leq \delta$.

1.3 Evidence for and Against

Despite significant work, the status of the Unique Games Conjecture — as well as the 2-to-1, 2-to-2, and α-Constraint Conjectures — is unresolved. Towards *disproving* the conjectures, the best algorithms known are due to Charikar, Makarychev, and Makarychev [2]. Using somewhat strong SDP relaxations, those authors gave polynomial-time SDP-rounding algorithms which achieve:

- Value $K^{-\epsilon/(2-\epsilon)}$ (roughly) for Unique Label-Cover instances with SDP value $1-\epsilon$ over alphabets of size K.
- Value $K^{-3+2\sqrt{2}-\epsilon}$ for 2-to-1 Label-Cover instances with SDP value $1-\Theta(\epsilon)$ over alphabets of size K.

The best evidence in *favor* of the Unique Games Conjecture is probably the existence of strong SDP gaps. The first such gap was given by Khot and Vishnoi [10]: they constructed a family of Unique Label-Cover instances over alphabet size K with SDP value $1-\epsilon$ and integral optimal value $K^{-\Theta(\epsilon)}$. In addition to roughly matching the CMM algorithm, the Khot–Vishnoi gaps have the nice property that they even hold with triangle inequality constraints added into the SDP. Even stronger SDP gaps for UGC were obtained recently by Raghavendra and Steurer [13].

Standing in stark contrast to this is the situation for the 2-to-1 Conjecture and related variants with perfect completeness. Prior to this work, there were *no known* SDP gap families for these problems with SDP value 1 and integral optimal value tending to 0 with the alphabet size. Indeed, there was hardly any evidence for these conjectures, beyond the fact that Charikar, Makarychev, and Makarychev failed to disprove them.

1.4 SDP Gaps as a Reduction Tool

In addition to being the only real evidence towards the validity of the UGC, SDP gaps for Unique Games have served another important role: they are the starting points for strong SDP gaps for other important optimization problems. A notable example of this comes in the work of Khot and Vishnoi [10] who used the UG gap instance to construct a super-constant integrality gap for the Sparsest Cut-SDP with triangle inequalities, thereby refuting the Goemans-Linial conjecture that the gap was bounded by $O(1)$. They also used this approach to show that the integrality gap of the Max-Cut SDP remains 0.878 when triangle inequalities are added. Indeed the approach via Unique Games remains the *only known* way to get such strong gaps for Max Cut. Recently, even stronger gaps for Max-Cut were shown using this framework in [9,13]. Another example of a basic problem

for which a SDP gap construction is only known via the reduction from Unique Games is Maximum Acyclic Subgraph [5].

In view of these results, it is fair to say that SDP gaps for Unique Games are significant unconditionally, regardless of the truth of the UGC. Given the importance of 2-to-1 and related conjectures in reductions to satisfiable CSPs and other problems like coloring where perfect completeness is crucial, SDP gaps for 2-to-1 Label-Cover and variants are worthy of study even beyond the motivation of garnering evidence towards the associated conjectures on their inapproximability.

2 Our Results

Label-Cover admits a natural semidefinite programming relaxation (see Figure 1). In this paper, we show the following results on the limitations of the basic semidefinite programming relaxation for Label-Cover instances with 2-to-1, 2-to-2, and α constraints:

- There is an instance of 2-to-2 Label-Cover with alphabet size K and optimum value $O(1/\log K)$ on which the SDP has value 1.
- There are instances of 2-to-1 and α-constraint Label-Cover with alphabet size K and optimum value $O(1/\sqrt{\log K})$ on which the SDP has value 1.

In both cases the instances have size $2^{\Omega(K)}$.

We note that if we only require the SDP value to be $1-\epsilon$ instead of 1, then integrality gaps for all these problems easily follow from gaps from Unique Games, constructed by Khot and Vishnoi [10] (by duplicating labels appropriately to modify the constraints). However, the motivation behind these conjectures is applications where it is important that the completeness is 1. Another difference between the 2-to-1 Label-Cover and the Unique Label-Cover is the fact that for 2-to-1 instances, it is consistent with known algorithmic results of [2] that OPT be as low as K^{-c} for some $c > 0$ independent of ϵ, when the SDP value is $1 - \epsilon$. It is an interesting question if OPT can indeed be this low even when the SDP value is 1. Our constructions do not address this question, as we only show $\mathrm{OPT} = O(1/\sqrt{\log K})$.

We also point out that our integrality gaps are for special cases of the Label-Cover problem where the constraints can be expressed as difference equations over $\mathbb{F}_2$-vector spaces. For example, for 2-to-2 Label-Cover, each constraint ϕ_e is of the form $x - y \in \{\alpha, \alpha + \gamma\}$ where $\alpha, \gamma \in \mathbb{F}_2^k$ are constants. For such constraints, the probable of deciding whether an instance is completely satisfiable ($\mathrm{OPT} = 1$) or not ($\mathrm{OPT} < 1$) is in fact in P. To see this, one can treat the coordinates $(x_1, \ldots, x_k)$ and $(y_1, \ldots, y_k)$ as separate boolean variables and introduce an auxiliary boolean variable z_e for each constraint. We can then rewrite the constraint as a conjunction of linear equations over $\mathbb{F}_2$:

$$\bigwedge_{i=1}^{k} (x_i - y_i - z_e \cdot \gamma_i = \alpha_i) .$$

Here $x_i, y_i, \alpha_i, \gamma_i$ denote the i^{th} coordinates of the corresponding vectors. Deciding whether a system of linear equations is completely satisfiable is of course in P. Alternatively, one can note that constraints $x - y \in \{\alpha, \alpha + \gamma\}$ mod $\mathbb{F}_2^k$ are *Mal'tsev constraints*, and hence deciding satisfiability of CSPs based on them is in P by the work of Bulatov and Dalmau [1].

Despite this tractability, the SDPs fail badly to decide satisfiability. This situation is similar to the very strong SDP gaps known for problems such as 3-XOR (see [14], [17]) for which deciding complete satisfiability is easy.

3 Preliminaries and Notation

3.1 Label-Cover Problems

In Figure 1, we write down a natural SDP relaxation for the Label-Cover problem. The relaxation is over the vector variables $\mathbf{z}_{(v,i)}$ for every vertex $v \in V$ and label $i \in [R]$.

$$\begin{array}{lll} \text{maximize} & \mathop{\mathbf{E}}_{e=(u,v)\in E}\Big[\sum_{i,j\in\psi_e} \big\langle \mathbf{z}_{(u,i)}, \mathbf{z}_{(v,j)} \big\rangle\Big] & \\ \text{subject to} & \sum_{i\in[R]} \big\|\mathbf{z}_{(v,i)}\big\|^2 = 1 & \forall\, v \in V \\ & \big\langle \mathbf{z}_{(v,i)}, \mathbf{z}_{(v,j)} \big\rangle = 0 & \forall\, i \neq j \in [R], v \in V \end{array}$$

Fig. 1. SDP for Label-Cover

Our goal in this work is to study integrality gaps for the above SDP for various special cases of the Label-Cover problem. We already discussed the Unique Games and 2-to-1 conjectures on the hardness of certain very special cases of Label-Cover. We now discuss two other variants of Label-Cover and their conjectured inapproximability.

Definition 3. *A constraint $\psi \subseteq \{1, \ldots, 2R\}^2$ is said to be a* 2-to-2 constraint *if there are two permutations σ_1, $\sigma_2 : \{1, \ldots, 2R\} \to \{1, \ldots, 2R\}$ such that $(i, j) \in \psi$ if and only if $(\sigma_1(i), \sigma_2(j)) \in T$ where*

$$T := \{(2l-1, 2l-1), (2l-1, 2l), (2l, 2l-1), (2l, 2l)\}_{l=1}^{R}.$$

A Label-Cover instance is said to be 2-to-2 if all its constraints are 2-to-2 constraints.

*A constraint $\psi \subseteq \{1, \ldots, 2R\}^2$ is said to be an α-*constraint *if there are two permutations σ_1, $\sigma_2 : \{1, \ldots, 2R\} \to \{1, \ldots, 2R\}$ such that $(i, j) \in \psi$ if and only if $(\sigma_1(i), \sigma_2(i)) \in T'$ where*

$$T' := \{(2l-1, 2l-1), (2l-1, 2l), (2l, 2l-1)\}_{l=1}^{R}.$$

A Label-Cover instance is said to be α if all its constraints are α constraints.

Conjecture 3. [3] (**2-to-2 Conjecture**) For any $\delta > 0$, it is NP-hard to decide whether a 2-to-2 Label-Cover instance $\mathcal{L}$ has $\mathrm{OPT}(\mathcal{L}) = 1$ or has $\mathrm{OPT}(\mathcal{L}) \leq \delta$.

It was shown in [3] that the 2-to-2 Conjecture is no stronger than the 2-to-1 Conjecture.

Conjecture 4. [3] (α **Conjecture**) For any $\delta > 0$, it is NP-hard to decide whether a α Label-Cover instance $\mathcal{L}$ has $\mathrm{OPT}(\mathcal{L}) = 1$ or has $\mathrm{OPT}(\mathcal{L}) \leq \delta$.

3.2 Fourier Analysis

Let $\mathcal{V} := \{f : \mathbb{F}_2^k \to \mathcal{R}\}$ denote the vector space of all real functions on $\mathbb{F}_2^k$, where addition is defined as point-wise addition. We always think of $\mathbb{F}_2^k$ as a probability space under the uniform distribution, and therefore use notation such as $\|f\|_p := \mathbf{E}_{x \in \mathbb{F}_2^k}[|f(x)|^p]$. For $f, g \in \mathcal{F}$, we also define the inner product $\langle f, g \rangle := \mathbf{E}[f(x)g(x)]$.

For any $\alpha \in \mathbb{F}_2^k$ the Fourier character $\chi_\alpha \in \mathcal{F}$ is defined by $\chi_\alpha(x) := (-1)^{\alpha \cdot x}$. The Fourier characters form an orthonormal basis for $\mathcal{V}$ with respect to the above inner product, hence every function $f \in \mathcal{V}$ has a unique representation as $f = \sum_{\alpha \in \mathbb{F}_2^k} \widehat{f}(\alpha)\chi_\alpha$, where the Fourier coefficient $\widehat{f}(\alpha) := \langle f, \chi_\alpha \rangle$.

We also sometimes identify each α with the set $S_\alpha = \{i \mid \alpha_i = 1\}$ and denote the Fourier coefficients as $\widehat{f}(S)$. We use the notation $|\alpha|$ for $|S_\alpha|$, the number of coordinates where α is 1.

The following well-known fact states that the norm of a function on $\mathbb{F}_2^k$ is unchanged when expressing it in the basis of the characters.

Proposition 1 (**Parseval's identity**). *For any* $f : \mathbb{F}_2^k \to \mathbb{R}$, $\sum_{\alpha \in \mathbb{F}_2^k} \widehat{f}(\alpha)^2 = \|f\|_2^2 = \mathbf{E}[f(x)^2]$.

We shall also need the following result due to Talagrand ("Proposition 2.3" in [15]), proven using hypercontractivity methods:

Theorem 2. *Suppose* $F : \mathbb{F}_2^k \to \mathbb{R}$ *has* $\mathbf{E}[F] = 0$. *Then*

$$\sum_{\alpha \in \mathbb{F}_2^k \setminus \{0\}} \widehat{F}(\alpha)^2/|\alpha| = O\left(\frac{\|F\|_2^2}{\ln(\|F\|_2/(e\|F\|_1))}\right).$$

More precisely, we will need the following easy corollary:

Corollary 1. *If* $F : \mathbb{F}_2^k \to \{0, 1\}$ *has mean* $1/K$, *then*

$$\widehat{F}(0)^2 + \sum_{\alpha \in \mathbb{F}_2^k \setminus \{0\}} \widehat{F}(\alpha)^2/|\alpha| = O\left(1/(K \log K)\right)$$

Proof. We have $\widehat{F}(0)^2 = \mathbf{E}[F]^2 = 1/K^2 \leq O(1/(K \log K))$, so we can disregard this term. As for the sum, we apply Theorem 2 to the function $F' = F - 1/K$, which has mean 0 as required for the theorem. It is easy to calculate that $\|F'\|_2 = \Theta(1/\sqrt{K})$ and $\|F'\|_1 = \Theta(1/K)$, and so the result follows.

4 Integrality Gap for 2-to-2 Games

We first give an integrality gap for label cover with 2-to-2 constraints. The instance for 2-to-1 label cover will be an extension of the one below. In fact, our analysis of OPT in the 2-to-1 case will follow simply by reducing it to the analysis of OPT for the 2-to-2 instance below.

The vertex set V in our instance is same as the vertex set of the Unique Games integrality gap instance constructed in [10]. Let $\mathcal{F} := \{f : \mathbb{F}_2^k \mapsto \{-1, 1\}\}$ denote the family of all boolean functions on $\mathbb{F}_2^k$. For $f,\ g \in \mathcal{F}$, define the product fg as $(fg)(x) := f(x)g(x)$. Consider the equivalence relation $\sim$ on $\mathcal{F}$ defined as $f \sim g \Leftrightarrow \exists \alpha \in \mathbb{F}_2^k\ s.t.\ f \equiv g\chi_\alpha$. This relation partitions $\mathcal{F}$ into equivalence classes $\mathcal{P}_1, \ldots, \mathcal{P}_n$, with $n := 2^K/K$. The vertex set V consists of the equivalence classes $\{\mathcal{P}_i\}_{i\in[n]}$. We denote by $[\mathcal{P}_i]$ the lexicographically smallest function in the class $\mathcal{P}_i$ and by $\mathcal{P}_f$, the class containing f.

We take the label set to be of size K and identify $[K]$ with $\mathbb{F}_2^k$ in the obvious way. For each tuple of the form (γ, f, g) where $\gamma \in \mathbb{F}_2^k \setminus \{0\}$ and $f, g \in \mathcal{F}$ are such that $(1 + \chi_\gamma)f \equiv (1 + \chi_\gamma)g$, we add a constraint $\psi_{(\gamma,f,g)}$ between the vertices $\mathcal{P}_f$ and $\mathcal{P}_g$. Note that the condition on f and g is equivalent to saying that $\chi_\gamma(x) = 1 \implies f(x) = g(x)$. If $f = [\mathcal{P}_f]\chi_\alpha$ and $g = [\mathcal{P}_g]\chi_\beta$ and if $A : [n] \to \mathbb{F}_2^k$ denotes the labeling, the relation $\psi_{(\gamma,f,g)}$ is defined as

$$(A(\mathcal{P}_f), A(\mathcal{P}_g)) \in \psi_{(\gamma,f,g)} \quad \Leftrightarrow \quad (A(\mathcal{P}_f) + \alpha) - (A(\mathcal{P}_g) + \beta) \ \in\ \{0, \gamma\}.$$

Note that for any $\omega \in \mathbb{F}_2^k$, the constraint maps the labels $\{\omega, \omega + \gamma\}$ for $\mathcal{P}_f$ to the labels $\{\omega+\alpha-\beta, \omega+\alpha-\beta+\gamma\}$ for $\mathcal{P}_g$ in a 2-to-2 fashion. We denote the set of all constraints by Ψ. We remark that, as in [10], our integrality gap instances contain multiple constraints on each pair of vertices.

4.1 SDP Solution

We give below a set of feasible vectors $\mathbf{z}_{(\mathcal{P}_i,\alpha)} \in \mathbb{R}^K$ for every equivalence class $\mathcal{P}_i$ and every label α, achieving SDP value 1. Identifying each coordinate with an $x \in \mathbb{F}_2^k$, we define the vectors as

$$\mathbf{z}_{(\mathcal{P}_i,\alpha)}(x) \ := \ \frac{1}{K}([\mathcal{P}_i]\chi_\alpha)(x).$$

It is easy to check that $\left\|\mathbf{z}_{(\mathcal{P}_i,\alpha)}\right\|^2 = 1/K$ for each of the vectors, which satisfies the first constraint. Also, $\mathbf{z}_{(\mathcal{P}_i,\alpha)}$ and $\mathbf{z}_{(\mathcal{P}_i,\beta)}$ are orthogonal for $\alpha \neq \beta$ since

$$\left\langle \mathbf{z}_{(\mathcal{P}_i,\alpha)}, \mathbf{z}_{(\mathcal{P}_i,\beta)} \right\rangle \ = \ \frac{1}{K^2} \left\langle [\mathcal{P}_i]\chi_\alpha, [\mathcal{P}_i]\chi_\beta \right\rangle \ = \ \frac{1}{K^2} \left\langle \chi_\alpha, \chi_\beta \right\rangle \ = \ 0$$

using the fact that $[\mathcal{P}_i]^2 = 1$. The following claim proves that the solution achieves SDP value 1.

Claim. For any edge e indexed by a tuple (γ, f, g) with $f(1 + \chi_\gamma) \equiv g(1 + \chi_\gamma)$, we have

$$\sum_{\omega_1,\omega_2\in\psi_{(\gamma,f,g)}} \left\langle \mathbf{z}_{(\mathcal{P}_f,\omega_1)}, \mathbf{z}_{(\mathcal{P}_g,\omega_2)} \right\rangle \ = \ 1$$

Proof. Let $f \equiv [\mathcal{P}_f]\chi_\alpha$ and $g \equiv [\mathcal{P}_g]\chi_\beta$. Then, $(\omega_1, \omega_2) \in \psi_e$ iff $(\omega_1 + \alpha) - (\omega_2 + \beta) \in \{0, \gamma\}$. Therefore, the above quantity equals (divided by 2 to account for double counting of ω)

$$\begin{aligned} & \frac{1}{2} \cdot \sum_{\omega} \Big(\big\langle \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha)}, \mathbf{z}_{(\mathcal{P}_g, \omega+\beta)} \big\rangle + \big\langle \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha+\gamma)}, \mathbf{z}_{(\mathcal{P}_g, \omega+\beta)} \big\rangle \\ & \qquad + \big\langle \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha)}, \mathbf{z}_{(\mathcal{P}_g, \omega+\beta+\gamma)} \big\rangle + \big\langle \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha+\gamma)}, \mathbf{z}_{(\mathcal{P}_g, \omega+\beta+\gamma)} \big\rangle \Big) \\ = \; & \frac{1}{2} \sum_{\omega} \big\langle \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha)} + \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha+\gamma)}, \mathbf{z}_{(\mathcal{P}_f, \omega+\beta)} + \mathbf{z}_{(\mathcal{P}_f, \omega+\beta+\gamma)} \big\rangle \end{aligned} \tag{1}$$

However, for each ω, we have $\mathbf{z}_{(\mathcal{P}_f, \omega+\alpha)} + \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha+\gamma)} = \mathbf{z}_{(\mathcal{P}_f, \omega+\beta)} + \mathbf{z}_{(\mathcal{P}_f, \omega+\beta+\gamma)}$, since for all coordinates x,

$$\begin{aligned} \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha)}(x) + \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha+\gamma)}(x) &= \frac{1}{K}([\mathcal{P}_f]\chi_{\omega+\alpha}(x) + [\mathcal{P}_f]\chi_{\omega+\alpha+\gamma}(x)) \\ &= \frac{1}{K}(f(x) + f\chi_\gamma)\chi_\omega(x) = \frac{1}{K}(g(x) + g\chi_\gamma)\chi_\omega(x) \\ = \frac{1}{K}([\mathcal{P}_g]\chi_{\omega+\beta}(x) + [\mathcal{P}_g]\chi_{\omega+\beta+\gamma}(x)) &= \mathbf{z}_{(\mathcal{P}_f, \omega+\beta)}(x) + \mathbf{z}_{(\mathcal{P}_f, \omega+\beta+\gamma)}(x). \end{aligned}$$

This completes the proof as the value of (1) then becomes

$$\frac{1}{2} \sum_{\omega} \left\| \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha)} + \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha+\gamma)} \right\|^2 = \frac{1}{2} \sum_{\omega} \left(\left\| \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha)} \right\|^2 + \left\| \mathbf{z}_{(\mathcal{P}_f, \omega+\alpha+\gamma)} \right\|^2 \right) = 1.$$

4.2 Soundness

We now prove that any labeling of the instance described above, satisfies at most $O(1/\log K)$ fraction of the constraints. Let $A : V \to \mathbb{F}_2^k$ be a labeling of the vertices. We extend it to a labeling of all the functions in $\mathcal{F}$ by defining $A([\mathcal{P}_i]\chi_\alpha) := A(\mathcal{P}_i) + \alpha$.

For each $\alpha \in \mathbb{F}_2^k$, define $A_\alpha : \mathcal{F} \to \{0, 1\}$ to be the indicator that A's value is α. By definition, the fraction of constraints satisfied by the labeling A is

$$\begin{aligned} \mathrm{val}(A) &= \mathop{\mathbf{E}}_{(\gamma, f, g) \in \Psi} \left[\sum_{\alpha \in \mathbb{F}_2^k} A_\alpha(f)(A_\alpha(g) + A_{\alpha+\gamma}(g)) \right] \\ &= \mathop{\mathbf{E}}_{(\gamma, f, g) \in \Psi} \left[\sum_{\alpha \in \mathbb{F}_2^k} A_\alpha(f)(A_\alpha(g) + A_\alpha(g\chi_\gamma)) \right] \\ &= 2 \cdot \mathop{\mathbf{E}}_{(\gamma, f, g) \in \Psi} \left[\sum_{\alpha \in \mathbb{F}_2^k} A_\alpha(f)(A_\alpha(g) \right] \end{aligned} \tag{2}$$

where the last equality used the fact that for every tuple $(\gamma, f, g) \in \Psi$, we also have $(\gamma, f, g\chi_\gamma) \in \Psi$.

Note that the extended labeling $A : \mathcal{F} \to \mathbb{F}_2^k$ takes on each value in $\mathbb{F}_2^k$ an equal number of times. Hence

$$\mathop{\mathbf{E}}_{f}[A_\alpha(f)] = \mathop{\mathbf{Pr}}_{f}[A(f) = \alpha] = 1/K \quad \text{for each } \alpha \in \mathbb{F}_2^k. \tag{3}$$

For our preliminary analysis, we will use only this fact to show that for any $\alpha \in \mathbb{F}_2^k$ it holds that

$$\mathop{\mathbf{E}}_{(\gamma,f,g)\in\Psi}[A_\alpha(f)A_\alpha(g)] \le O(1/(K \log K)). \tag{4}$$

It will then follow that the soundness (2) is at most $O(1/\log K)$. Although this tends to 0, it does so only at a rate proportional to the logarithm of the alphabet size, which is $K = 2^k$.

Beginning with the left-hand side of (4), let's write $F = A_\alpha$ for simplicity. We think of the functions f and g being chosen as follows. We first choose a function $h : \gamma^\perp \to \{-1, 1\}$. Note that $\gamma^\perp \subseteq \mathbb{F}_2^k$ is the set of inputs where $\chi_\gamma = 1$ and hence $f = g$, and we let $f(x) = g(x) = h(x)$ for $x \in \gamma_\perp$. The values of f and g on the remaining inputs are chosen independently at random. Then

$$\begin{aligned}\mathop{\mathbf{E}}_{(\gamma,f,g)\in\Psi}[F(f)F(g)] &= \mathop{\mathbf{E}}_{\gamma}\mathop{\mathbf{E}}_{h:\gamma^\perp\to\{-1,1\}}\left[\mathop{\mathbf{E}}_{f,g|h}[F(f)F(g)]\right] \\ &= \mathop{\mathbf{E}}_{\gamma}\mathop{\mathbf{E}}_{h:\gamma^\perp\to\{-1,1\}}\left[\mathop{\mathbf{E}}_{f|h}[F(f)]\mathop{\mathbf{E}}_{g|h}[F(g)]\right]. \end{aligned}\tag{5}$$

Let us write $P_\gamma F(h)$ for $\mathbf{E}_{f|h} F(f)$, which is also equal to $\mathbf{E}_{g|h} F(g)$. We now use the Fourier expansion of F. Note that the domain here is $\{-1, 1\}^K$ instead of $\mathbb{F}_2^k$. To avoid confusion with characters and Fourier coefficients for functions on $\mathbb{F}_2^k$, we will index the Fourier coefficients below by sets $S \subseteq \mathbb{F}_2^k$. Given an $f \in V$, we'll write f^S for $\prod_{x\in S} f(x)$ (which is a Fourier character for the domain $\{-1, 1\}^K$). Now for fixed γ and h,

$$P_\gamma F(h) = \mathop{\mathbf{E}}_{f|h}[F(f)] = \mathop{\mathbf{E}}_{f|h}\left[\sum_{S\subseteq\mathbb{F}_2^k}\widehat{F}(S)f^S\right] = \sum_{S\subseteq\mathbb{F}_2^k}\widehat{F}(S)\cdot\mathop{\mathbf{E}}_{f|h}[f^S].$$

The quantity $\mathbf{E}_{f|h}[f^S]$ is equal to h^S if $S \subseteq \gamma^\perp$ as is 0 otherwise. Thus, using the Parseval identity, we deduce that (5) equals

$$\mathop{\mathbf{E}}_{\gamma}\mathop{\mathbf{E}}_{h:\gamma^\perp\to\{-1,1\}}\left[(P_\gamma F(h))^2\right] = \mathop{\mathbf{E}}_{\gamma}\left[\sum_{S\subseteq\gamma^\perp}\left(\widehat{F}(S)\right)^2\right] = \sum_{S\subseteq\mathbb{F}_2^k}\mathop{\mathbf{Pr}}_{\gamma}[S \subseteq \gamma^\perp]\cdot\left(\widehat{F}(S)\right)^2.$$

Recalling that $\gamma \in \mathbb{F}_2^k \setminus \{0\}$ is chosen uniformly, we have that

$$\sum_{S\subseteq\mathbb{F}_2^k}\mathop{\mathbf{Pr}}_{\gamma}[S \subseteq \gamma^\perp]\cdot\left(\widehat{F}(S)\right)^2 = \sum_{S\subseteq\mathbb{F}_2^k} 2^{-\dim(S)}\cdot\left(\widehat{F}(S)\right)^2,$$

where we are writing $\dim(S) = \dim(\text{span } S)$ for shortness (and defining $\dim(\emptyset) = 0$). For $|S| \geq 1$ we have $\dim(S) \geq \log_2 |S|$ and hence $2^{-\dim(S)} \geq 1/|S|$. Thus

$$\sum_{S \subseteq \mathbb{F}_2^k} 2^{-\dim(S)} \cdot \widehat{F}(S)^2 \leq \widehat{F}(\emptyset)^2 + \sum_{\emptyset \neq S \subseteq \mathbb{F}_2^k} \widehat{F}(S)^2/|S|.$$

Corollary 1 shows that this is at most $O(1/(K \log K))$. This completes the proof:

$$\text{val}(A) = 2 \cdot \sum_{\alpha \in \mathbb{F}_2^k} \mathop{\mathbf{E}}_{(\gamma,f,g) \in \Psi} [A_\alpha(f) A_\alpha(g)] \;\leq\; 2 \cdot \sum_{\alpha \in \mathbb{F}_2^k} 2^{-\dim(S)} \widehat{A_\alpha}(S)^2 = O(1/\log K).$$

5 Integrality Gap for 2-to-1 Label Cover

The instances for 2-to-1 label cover are bipartite. We denote such instances as (U, V, E, R_1, R_2, Π) where $R_2 = 2R_1$ denote the alphabet sizes on the two sides. Due to lack of space, in this extended abstract we only state our integrality gap instances. In the full version of this work we prove that these instances have SDP value 1 and OPT value $O(1/\sqrt{\log K})$, by reduction to our 2-to-2 gap analysis.

As in the case of 2-to-2 games, the set V consists of equivalence classes $\mathcal{P}_1, \ldots, \mathcal{P}_n$, which partition the set of functions $\mathcal{F} = \{f : \mathbb{F}_2^k \to \{-1, 1\}\}$, according to the equivalence relation $\sim$ defined as $f \sim g \;\Leftrightarrow\; \exists \alpha \in \mathbb{F}_2^k \; s.t. f \equiv g\chi_\alpha$. The label set $[R_2]$ is again identified with $\mathbb{F}_2^k$ and is of size $K = 2^k$.

To describe the set U, we further partition the vertices in V according to other equivalence relations. For each $\gamma \in \mathbb{F}_2^k, \gamma \neq 0$, we define an equivalence relation $\cong_\gamma$ on the set $\mathcal{P}_1, \ldots, \mathcal{P}_n$ as

$$\mathcal{P}_i \cong_\gamma \mathcal{P}_j \qquad \Leftrightarrow \qquad \exists f \in \mathcal{P}_i, g \in \mathcal{P}_j \text{ s.t. } f(1 + \chi_\gamma) \equiv g(1 + \chi_\gamma).$$

This partitions $\mathcal{P}_1, \ldots, \mathcal{P}_n$ (and hence also the set $\mathcal{F}$) into equivalence classes $\mathcal{Q}_1^\gamma, \ldots, \mathcal{Q}_m^\gamma$. It can be verified that the value of m turns out to be $2^{K/2+1}/K$ and the partition is different for each γ. The set U has one vertex for each class of the form $\mathcal{Q}_i^\gamma$ for all $i \in [m]$ and $\gamma \in \mathbb{F}_2^k \setminus \{0\}$. As before, we denote by $[\mathcal{Q}_i^\gamma]$ the lexicographically smallest function in the class $\mathcal{Q}_i^\gamma$, and by $\mathcal{Q}_f^\gamma$ the class under $\cong_\gamma$ containing f. Note that if $f \in \mathcal{Q}_i^\gamma$, then there exists a $\beta \in \mathbb{F}_2^k$ such that $f(1 + \chi_\gamma) \equiv [\mathcal{Q}_i^\gamma]\chi_\beta(1 + \chi_\gamma)$.

The label set R_1 has size $K/2$. For each vertex $\mathcal{Q}_i^\gamma \in U$, we think of the labels as pairs of the form $\{\alpha, \alpha + \gamma\}$ for $\alpha \in \mathbb{F}_2^k$. More formally, we identify it with the space $\mathbb{F}_2^k / \langle\gamma\rangle$. We impose one constraint for every pair of the form (γ, f) between the vertices $\mathcal{P}_f$ and $\mathcal{Q}_f^\gamma$. If $f \equiv [\mathcal{P}_f]\chi_\alpha$ and $f(1 + \chi_\gamma) \equiv [\mathcal{Q}_i^\gamma]\chi_\beta(1 + \chi_\gamma)$, then the corresponding relation $\psi_{(\gamma,f)}$ is defined by requiring that for any labelings $A : V \to [R_2]$ and $B : U \to [R_1]$,

$$(B(\mathcal{Q}_f^\gamma), A(\mathcal{P}_f)) \in \psi_{(\gamma,f)} \quad \Leftrightarrow \quad A(\mathcal{P}_f) + \alpha \;\in\; B(\mathcal{Q}_f^\gamma) + \beta.$$

Here, if $B(\mathcal{Q}_f^\gamma)$ is a pair of the form $\{\omega, \omega + \gamma\}$, then $B(\mathcal{Q}_f^\gamma) + \beta$ denotes the pair $\{\omega + \beta, \omega + \gamma + \beta\}$.

6 From 2-to-1 Constraints to α-Constraints

In the full version of the paper, we that any integrality gap instance for 2-to-1 games with sufficiently many edges can be converted to an integrality gap instance for games with α-constraints. The SDP we consider for these games is identical to the ones considered before, except for the objective function.

Theorem 3. *Let $\mathcal{L} = (U, V, E, R, 2R, \Psi)$ be a bipartite instance of 2-to-1 label cover problem with $\mathrm{OPT}(\mathcal{L}) \leq \delta$ and SDP value 1. Also, let $|E| \geq 4(|U| + |V|)\log(R)/\epsilon^2$. Then there exists another instance $\mathcal{L}' = (U, V, E, 2R, \Psi')$ of Label Cover with α-constraints having SDP value 1 and $\mathrm{OPT}(\mathcal{L}') \leq \delta + \epsilon + 1/R$.*

In brief, the proof of this result is by adding R "fake" labels for each vertex $u \in U$, and then randomly augmenting the constraints to make them of the required form.

7 Discussion

The instances we construct have SDP value 1 only for the most basic semidefinite programming relaxation. It would be desirable to get gaps for stronger SDPs, beginning with the most modest extensions of this basic SDP. For example, in the SDP for 2-to-1 Label Cover from Figure 1, we can add valid nonnegativity constraints for the dot product between every pair of vectors in the set

$$\{\mathbf{z}_{(u,i)} \mid u \in U, i \in [R]\} \ ,$$

since in the integral solution all these vectors are $\{0,1\}$-valued. The vectors we construct do *not* obey such a nonnegativity requirement. For the case of Unique Games, Khot and Vishnoi [10] were able to ensure nonnegativity of all dot products by taking tensor products of the vectors with themselves and defining new vectors $\mathbf{y}'_{(u,i)} = \mathbf{y}^{\otimes 2}_{(u,i)} = \mathbf{y}_{(u,i)} \otimes \mathbf{y}_{(u,i)}$ and $\mathbf{z}'_{(v,j)} = \mathbf{z}^{\otimes 2}_{(v,j)} = \mathbf{z}_{(v,j)} \otimes \mathbf{z}_{(v,j)}$. Since $\langle \mathbf{a}^{\otimes 2}, \mathbf{b}^{\otimes 2} \rangle = \langle \mathbf{a}, \mathbf{b} \rangle^2$, the desired nonnegativity of dot products is ensured.

We cannot apply this tensoring idea in our construction as it does not preserve the SDP value at 1. For example, for 2-to-2 Label Cover, if we have $\mathbf{z}_{(u,i_1)} + \mathbf{z}_{(u,i_2)} = \mathbf{z}_{(v,j_1)} + \mathbf{z}_{(v,j_2)}$ (so that these vectors contribute 1 to the objective value to the SDP of Figure 1), then upon tensoring we no longer necessarily have $\mathbf{z}^{\otimes 2}_{(u,i_1)} + \mathbf{z}^{\otimes 2}_{(u,i_2)} = \mathbf{z}^{\otimes 2}_{(v,j_1)} + \mathbf{z}^{\otimes 2}_{(v,j_2)}$. Extending our gap instances to obey the nonnegative dot product constraints is therefore a natural question that we leave open. While this seems already quite challenging, one can of course be more ambitious and ask for gap instances for stronger SDPs that correspond to certain number of rounds of some hierarchy, such as the Sherali-Adams hierarchy together with consistency of vector dot products with pairwise marginals. For Unique Games, gap instances for several rounds of such a hierarchy were constructed in [13].

Acknowledgments

Madhur would like to thank Per Austrin for helpful discussions in the early stages of this work. We also thank the anonymous reviewers for their suggestions.

References

1. Bulatov, A., Dalmau, V.: A simple algorithm for Mal'tsev constraints. SICOMP 36(1), 16–27 (2006)
2. Charikar, M., Makarychev, K., Makarychev, Y.: Near-optimal algorithms for unique games. In: Proc. 38^{th} ACM STOC, pp. 205–214 (2006)
3. Dinur, I., Mossel, E., Regev, O.: Conditional hardness for approximate coloring. SICOMP 39(3), 843–873 (2009)
4. Dinur, I., Safra, S.: On the hardness of approximating minimum vertex cover. Ann. Math. 162(1), 439–486 (2005)
5. Guruswami, V., Manokaran, R., Raghavendra, P.: Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In: Proc. 49^{th} IEEE FOCS, pp. 573–582 (2008)
6. Guruswami, V., Sinop, A.K.: Improved inapproximability results for maximum k-colorable subgraph. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX 2009 and RANDOM 2009. LNCS, vol. 5687, pp. 163–176. Springer, Heidelberg (2009)
7. Khot, S.: Hardness results for coloring 3-colorable 3-uniform hypergraphs. In: Proc. 43^{rd} IEEE FOCS, pp. 23–32 (2002)
8. Khot, S.: On the power of unique 2-prover 1-round games. In: Proc. 34^{th} ACM STOC, pp. 767–775 (2002)
9. Khot, S., Saket, R.: SDP integrality gaps with local ℓ_1-embeddability. In: Proc. 50^{th} IEEE FOCS, pp. 565–574 (2009)
10. Khot, S., Vishnoi, N.: The Unique Games Conjecture, integrality gap for cut problems and embeddability of negative type metrics into ℓ_1. In: Proc. 46^{th} IEEE FOCS, pp. 53–62 (2005)
11. O'Donnell, R., Wu, Y.: Conditional hardness for satisfiable CSPs. In: Proc. 41^{st} ACM STOC, pp. 493–502 (2009)
12. Raghavendra, P.: Optimal algorithms and inapproximability results for every CSP? In: Proc. 40^{th} ACM STOC, pp. 245–254 (2008)
13. Raghavendra, P., Steurer, D.: Integrality gaps for strong SDP relaxations of unique games. In: Proc. 50^{th} IEEE FOCS, pp. 575–585 (2009)
14. Schoenebeck, G.: Linear level Lasserre lower bounds for certain k-CSPs. In: Proc. 49^{th} IEEE FOCS, pp. 593–602 (2008)
15. Talagrand, M.: On Russo's approximate zero-one law. Ann. Prob. 22(3), 1576–1587 (1994)
16. Tang, L.: Conditional hardness of approximating satisfiable Max 3CSP-q. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 923–932. Springer, Heidelberg (2009)
17. Tulsiani, M.: CSP gaps and reductions in the Lasserre hierarchy. In: Proc. 41^{st} ACM STOC, pp. 303–312 (2009)

Data Stream Algorithms for Codeword Testing*
(Extended Abstract)

Atri Rudra and Steve Uurtamo

Department of Computer Science and Engineering,
University at Buffalo, The State University of New York,
Buffalo, NY, 14620
{atri,uurtamo}@buffalo.edu

Abstract. Motivated by applications in storage systems and property testing, we study data stream algorithms for local testing and tolerant testing of codes. Ideally, we would like to know whether there exist asymptotically good codes that can be local/tolerant tested with one-pass, poly-log space data stream algorithms.

We show that for the error detection problem (and hence, the local testing problem), there exists a one-pass, log-space data stream algorithm for a broad class of asymptotically good codes, including the Reed-Solomon (RS) code and expander codes. In our technically more involved result, we give a one-pass, $O(e \log^2 n)$-space algorithm for RS (and related) codes with dimension k and block length n that can distinguish between the cases when the Hamming distance between the received word and the code is at most e and at least $a \cdot e$ for some absolute constant $a > 1$. For RS codes with random errors, we can obtain $e \leq O(n/k)$. For folded RS codes, we obtain similar results for worst-case errors as long as $e \leq (n/k)^{1-\varepsilon}$ for any constant $\varepsilon > 0$. These results follow by reducing the tolerant testing problem to the error detection problem using results from group testing and the list decodability of the code. We also show that using our techniques, the space requirement and the upper bound of $e \leq O(n/k)$ cannot be improved by more than logarithmic factors.

1 Introduction

In this work, we consider data stream algorithms for local testing and tolerant testing of error-correcting codes. The local testing problem for a code $C \subseteq \Sigma^n$ is the following: given a received word $\boldsymbol{y} \in \Sigma^n$, we need to figure out if $\boldsymbol{y} \in C$ or if $\boldsymbol{y}$ differs from every codeword in C in at least $0 < e \leq n$ positions (i.e. the Hamming distance of $\boldsymbol{y}$ from every $\boldsymbol{c} \in C$, denoted by $\Delta(\boldsymbol{y}, \boldsymbol{c})$, is at least e). If $e = 1$, then this is the error-detection problem. In the tolerant testing problem, given $\boldsymbol{y}$, we need to decide if $\boldsymbol{y}$ is at a distance at most e_1 from some codeword or if it has distance at least $e_2 > e_1$ from every codeword in C. Ideally, we would like to answer the following (see Section 2 for definitions related to codes):

* Research supported by NSF CAREER Award CCF-0844796.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 629–640, 2010.

Question 1. *Do there exist asymptotically good codes that can be (tolerant) tested by one-pass, poly-log space data stream algorithms?*

To the best of our knowledge, ours is the first work that considers this natural problem. We begin with the motivation for our work.

Property Testing. Local testing of codes has been extensively studied under the stricter requirements of property testing. Under the property testing requirements, one needs to solve the local testing problem by (ideally) only accessing a constant number of positions in $\boldsymbol{y}$. Codes that can be locally tested with a constant number of queries have been instrumental in the development of the PCP machinery, starting with the original proof of the PCP theorem [1,2]. The current record is due to Dinur, who presents codes that have inverse poly-log rate, linear distance and can be locally tested with a constant number of queries [9].

General lower bounds on local testing of codes, however, have been scarce. (See, e.g. the recent paper [5]). In particular, it is not known if there are asymptotically good codes that can be locally tested with a constant number of queries. The question remains open even if one considers the harder task of tolerant testing with a constant number of *non-adaptive* queries [13].

It is not too hard to see that a non-adaptive tolerant tester that makes a constant number of queries gives a single pass, log-space data stream algorithm for tolerant testing [17]. Thus, if one could prove that any one-pass data stream algorithm for local/tolerant testing of asymptotically good codes requires $\omega(\log n)$ space, then they will have answered the question in the negative (at least for non-adaptive queries). This could present a new approach to attack the question of local/tolerant testing with a constant number of queries.

Next, we discuss the implications of *positive* results for local/tolerant testing.

Applications in Storage Systems. Codes are used in current day storage systems such as optical storage (CDs and DVDs), RAID ([8]) and ECC memory ([7]). Storage systems, up until recently, used simple codes such as the parity code and checksums, which have (trivial) data stream algorithms for error detection. However, the parity code cannot detect even two errors. With the explosion in the amount of data that needs to be stored, errors are becoming more frequent. This situation will become worse as more data gets stored on disks [11]. Thus, we need to use codes that can handle more errors.

Reed-Solomon (RS) codes, which are used widely in storage systems (e.g. in CDs and DVDs and more recently in RAID), are well known to have good error correcting capabilities. However, the conventional error detection for RS codes is not space or pass efficient. Thus, a natural question to ask is if one can design a data stream algorithm to perform error detection for RS codes.

It would be remiss of us not to point out that unlike a typical application of a data stream algorithm where n is very large, in real life deployments of RS codes, n is relatively small. However, if one needs to implement the error detection algorithm in controllers on disks then it would be advantageous to use a data stream algorithm so that it is feasible to perform error detection with every read. Another way to use error detection is in *data scrubbing* [11].

Here, error detection is run on the entire disk to catch errors. In addition, the single pass requirement means that we will probe each bit on a disk only once, which is good for longevity of data. Finally, it would be useful to generalize an error-detector to a data stream algorithm that could also *locate* the errors.

It is also plausible that the efficiency of the data stream algorithms will make it feasible to use RS codes of block length (and alphabet size) considerably larger than the ones currently in use. For the storage application, designing algorithms for a widely used code (such as RS) will be more valuable than answering Question 1 in the affirmative via some new code. Also, even solving the tolerant testing problem in this case with large constants e_1 and e_2 would be useful. (Although local testing a RS code of dimension k requires at least k queries[1], it does not rule out the possibility of a tester in a data stream setting).

Our Results. We give a one-pass, poly-log space, randomized algorithm to perform error detection for a broad class of asymptotically good codes such as Reed-Solomon (RS) and expander codes. As a complementary result, we also show that deterministic data stream algorithms (even with multiple passes) require linear space for such codes. Thus, for local testing we answer Question 1 in the affirmative. This should be contrasted with the situation in property testing, where it is known that for both asymptotically good RS and expander codes, a linear number of queries is required. (The lower bound for RS codes was discussed in the paragraph above and the result for expander codes follows from [6].)

It turns out that using existing results for tolerant testing of Reed-Muller codes over large alphabets [13], one can answer Question 1 in the affirmative for tolerant testing, though with $O(n^{\varepsilon})$ space for any constant $\varepsilon > 0$ [17].

Given the practical importance of RS codes and given the fact that local testing for RS codes with data stream constraints is possible, for the rest of the paper we focus mostly on tolerant testing of RS and related codes. We first remark that a naive tolerant testing algorithm for RS codes that can be implemented in $O(e \log n)$ space is to go through all the $\sum_{i=1}^{e} \binom{n}{i}$ possible error locations S and check if the received word projected outside of S belongs to the corresponding projected down RS code. (This works as long as $e \leq n-k$, which is true w.l.o.g. since $n-k$ is the covering radius of a RS code of dimension k and block length n.) Using our error detector for RS codes, this can be implemented as a one-pass $O(e \log n)$-space data stream algorithm. Unfortunately, the run time of this algorithm is prohibitive, even for moderate values of e.

In this paper, we match the parameters above to within a log factor but with a (small) polynomial running time for values of e much larger than a constant. In particular, we present a one-pass, $O(e \log^2 n)$-space, polynomial time randomized algorithm for a RS code C with dimension k and block length n that can distinguish between the cases when the Hamming distance between $\boldsymbol{y}$ and C is at most e and at least $a \cdot e$ (for some constant $a > 1$). This reduction works when $e(e+k) \leq O(n)$. If we are dealing with random errors, then we can obtain $ek \leq O(n)$. Using known results on list decodability of folded RS codes [14], we

[1] This follows from the fact that the "dual" code has distance $k+1$.

obtain similar results for worst case errors for $e \leq (n/k)^{1-\varepsilon}$ for any constant $\varepsilon > 0$. As a byproduct, our algorithms also locate the errors (if the number of errors is bounded by e), which is desirable for the storage application. We also show that using our techniques, the space requirement and the upper bound of $e \leq O(n/k)$ cannot be improved by more than logarithmic factors.

Ideally, we would like our data stream algorithms to spend poly-log time per input position. However, in this paper we will tolerate polynomial time algorithms. In particular, naive implementations of the tolerant testing algorithms take $\tilde{O}(n^2)$ time. We also show that at the expense of slightly worse parameters, we can achieve a running time of $\tilde{O}(ne)$ for certain RS codes.

Our Techniques. It is well known that error detection for any linear code can be done by checking if the product of the received word with the parity check matrix is the all zeros vector. We turn this into a one-pass low space data stream algorithm using the fingerprinting method. The only difference from the usual fingerprinting method, where one uses any large enough field, is that we need to use a large enough extension field of the finite field over which the code is defined. To show the necessity of randomness, we use the fooling set method from communication complexity [16]. However, unlike the usual application of two-party communication complexity in data stream algorithms, where the stream is broken up into two fixed portions, in our case we need to be careful about how we divide up the input [17].

We now move on to our tolerant testing algorithm. We begin with the connection to group testing. Let $\boldsymbol{c}$ be the closest codeword to $\boldsymbol{y}$ and let $\boldsymbol{x} \in \{0,1\}^n$ denote the binary vector where $x_i = 1$ iff $y_i \neq c_i$. Now assume we could access $\boldsymbol{x}$ in the following manner: pick a subset $Q \subseteq [n]$ and check if $\boldsymbol{x}_Q = \mathbf{0}$ or not (where $\boldsymbol{x}_Q$ denotes $\boldsymbol{x}$ projected onto indices in Q). Then can we come up with a clever way of non-adaptively choosing the tests such that at the end we know whether $\textsc{wt}(\boldsymbol{x}) \leq e$ or not? It turns out that we can use group testing to construct such an algorithm. In fact, using e-disjunct matrices (cf. [10]), we can design non-adaptive tests such that given the answers to the tests one could compute $\boldsymbol{x}$ if $\textsc{wt}(\boldsymbol{x}) \leq e$, else determine that $\textsc{wt}(\boldsymbol{x}) > e$. (A natural question is how many tests do e-disjunct matrices require: we will come back to this question in a bit.) This seems to let us test whether $\boldsymbol{y}$ is within a Hamming distance of e from some codeword or not. Note that the above is essentially reducing one instance of the tolerant testing problem to multiple instances of error-detection.

Thus, all we need to do is come up with a way to implement the tests to $\boldsymbol{x}$. A natural option, which we take, is that for any test $Q \subseteq [n]$, we check if $\boldsymbol{y}_Q \in \mathrm{RS}_Q[k]$, where $\mathrm{RS}_Q[k]$ is the RS code (of dimension k) projected onto Q. This immediately puts in one restriction: we will need $|Q| \geq k$ (as otherwise every test will return a yes). However, there is a subtle issue that makes our analysis more complicated– we do not necessarily have that $\boldsymbol{x}_Q = \mathbf{0}$ iff $\boldsymbol{y}_Q \in \mathrm{RS}_Q[k]$. While it is true that $\boldsymbol{x}_Q = 0$ implies $\boldsymbol{y}_Q \in RS_Q[k]$, the other direction is not true. The latter is possible only if $\boldsymbol{y}$ agrees with some codeword $\boldsymbol{c}' \neq \boldsymbol{c}$ in the positions indexed by Q. Now if s is the size of the smallest test and it is the case that the only codeword that agrees with $\boldsymbol{y}$ in at least s positions is $\boldsymbol{c}$, then

we'll be done. We show that this condition is true for RS codes if $s \geq e + k$ for adversarial errors or with high probability if $s \geq 4k$ for random errors.

It is now perhaps not surprising that the list decodability of the code plays a role in our general result for worst-case errors. Assume that the code C under consideration is $(n-s, L)$ list decodable (i.e. every Hamming ball of radius $n-s$ has at most L codewords in it) and one can do error detection on C projected down to any test of size at least s. If we pick our disjunct matrix carefully and L is not too large, it seems intuitive that one should be able to have, for most of the tests, that $\boldsymbol{x}_Q \neq \mathbf{0}$ implies $\boldsymbol{y}_Q \notin C_Q$. We are able to show that if the matrix is picked at random, then this property holds. In addition, it is the case that the "decoding" of $\boldsymbol{x}$ from the result of the test can be done even if some of the test results are faulty (i.e. $\boldsymbol{y}_Q \in C_Q$ even though $\boldsymbol{x}_Q \neq \mathbf{0}$). The proof of this fact requires a fair bit of work: we will come back to the issues in a bit.

Another side-effect of the fact that our algorithm does not readily translate into the group testing scenario is that even though we have been able to salvage the case for $\text{WT}(\boldsymbol{x}) \leq e$, we can no longer guarantee that if $\text{WT}(\boldsymbol{x}) > e$, that our algorithm will catch it. In the latter case, our algorithm might say $\text{WT}(\boldsymbol{x}) > e$ or it might return a subset $S \subseteq [n]$ that purportedly contains all the error locations. However, we can always check if $\boldsymbol{y}_{[n]\setminus S} \in \text{RS}_{[n]\setminus S}[k]$ to rule out the latter case. This seems to require another pass on the input but we are able to implement the final algorithm in one pass by giving a one-pass algorithm for the following problem: Given as input $\boldsymbol{y}$ followed by $T \subseteq [n]$ such that $|T| = e$, design a one-pass $O(e \log n)$ space algorithm to check if $\boldsymbol{y}_{[n]\setminus T} \in \text{RS}_{[n]\setminus T}[k]$. The main idea is to encode the locations in T as an unknown degree e polynomial and to fill in the unknown coefficients once the algorithm gets to T in the input.

We now return to the question of how many tests are needed for e-disjunct matrices. The best known construction uses $O(e^2 \log n)$ tests [10] and this is tight to within a $\log e$ factor (cf. [12]). Thus, to get sublinear space, we need to have $e = o(\sqrt{n})$. To break the $\sqrt{n}$ barrier, instead of e-disjunct matrices, we use the notion of (e, e)-list disjunct matrix [15]. An (e, e)-list disjunct matrix has the property that when applied to $\boldsymbol{x}$ such that $\text{WT}(\boldsymbol{x}) \leq e$, it returns a subset $S \subseteq [n]$ such that (i) $x_i = 1$ implies $i \in S$ and (ii) $|S| \leq \text{WT}(\boldsymbol{x}) + e$. It is known that such matrices exist with $O(e \log n)$ rows. We show that such matrices can be constructed with $O(e \log^2 n)$ random bits. However, note we can now only distinguish between the cases of $\text{WT}(\boldsymbol{x}) \leq e$ and $\text{WT}(\boldsymbol{x}) \geq 2e$.

The use of list disjunct matrices also complicates our result for worst case errors that uses the list decodability of the code under consideration. The issue is that when we pick the desired matrix at random, with the extra task of "avoiding" all of the $L-1$ codewords other than $\boldsymbol{c}$ that can falsify the answer to the test, we can only guarantee that the "decoding" procedure is able to recover a constant fraction of the positions in error. This is similar to the notion of error reduction in [19]. This suggests a natural, iterative $O(\log e)$-pass algorithm. Using our earlier trick, we can again implement our algorithm in one pass. Finally, the plain vanilla proof needs $\Omega(n)$ random bits. We observe that the proof goes through with limited independence and use this to reduce the amount of

randomness to $O(e^2 \log^3 n)$ bits. Reducing the random bits to something smaller, such as $O(e \log n)$, is an open problem.

The speedup in the run time from the naive $\tilde{O}(n^2)$ to $\tilde{O}(ne)$ for the tolerant testing algorithms is obtained by looking at certain explicit disjunct matrices and observing that the reduced error detection problems are nicely structured.

There are two unsatisfactory aspects of our algorithms: (i) The $O(e \log^2 n)$ space complexity and (ii) The condition that $e \leq O(n/k)$ (which in turn follows from the fact that we have $s = n/(2e)$). We show that both of these shortcomings are essentially unavoidable with our techniques. In particular, a lower bound on the 1^+ decision tree complexity of the threshold function from [3] implies that at least $\Omega(e)$ invocations of the error detection routine are needed. Further, we show that for sublinear test complexity, the support size s must be in $O(\frac{n}{e} \log n)$. This follows by interpreting the reduction as a set cover problem and observing that any set covers only a very small fraction of the universe.

Due to space restrictions all proofs and some discussions are omitted from this extended abstract. The details can be found in the full version [17].

2 Preliminaries

We begin with some notation. Given an integer m, we will use $[m]$ to denote the set $\{1, \ldots, m\}$. We will denote by $\mathbb{F}_q$ the finite field with q elements. An $a \times b$ matrix M over $\mathbb{F}_q$ will be called strongly explicit if given any $(i, j) \in [a] \times [b]$, the entry $M_{i,j}$ can be computed in space poly$(\log q + \log a + \log b)$. Given a vector $\boldsymbol{y} \in \Sigma^n$ ($C \subseteq \Sigma^n$ resp.) and a subset $S \subseteq [n]$, we will use $\boldsymbol{y}_S$ (C_S resp.) to denote $\boldsymbol{y}$ (vectors in C resp.) projected down to the indices in S. We will use $\text{WT}(\boldsymbol{x})$ to denote the number of non-zero entries in $\boldsymbol{x}$. Further, for $S \subseteq [n]$, we will use $\text{WT}_S(\boldsymbol{x})$ to denote $\text{WT}(\boldsymbol{x}_S)$.

Codes. A code of *dimension* k and *block length* n over an alphabet Σ is a subset of Σ^n of size $|\Sigma|^k$. The *rate* of such a code equals k/n. A code C over $\mathbb{F}_q$ is called a linear code if C is a linear subspace of $\mathbb{F}_q^n$. If C is linear, then it can be described by its parity-check matrix H, i.e. for every $\boldsymbol{c} \in C$, $H \cdot \boldsymbol{c}^T = \boldsymbol{0}$. An asymptotically good code has constant rate and constant relative distance (i.e. any two codewords differ in at least some fixed constant fraction of positions).

Tolerant Testers. We begin with the central definition. Given a code $C \subseteq \Sigma^n$, reals $0 \leq d < c \leq 1$, $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$ and integers $r = r(n)$ and $s = s(n)$, an $(r, s, \varepsilon_1, \varepsilon_2)_{c,d}$-tolerant tester $\mathcal{T}$ for C is a randomized algorithm with the following properties for any input $\boldsymbol{y} \in \Sigma^n$: (1) If $\Delta(\boldsymbol{y}, C) \leq \varepsilon_1 n$, then $\mathcal{T}$ accepts with probability at least c; (2) If $\Delta(\boldsymbol{y}, C) \geq \varepsilon_2 n$, then $\mathcal{T}$ accepts with probability at most d; (3) $\mathcal{T}$ makes at most r passes over $\boldsymbol{y}$; and (4) $\mathcal{T}$ uses at most s space for its computation.

Further, we will consider the following special cases of an $(r, s, \varepsilon_1, \varepsilon_2)_{c,d}$-tolerant tester: (i) An $(r, s, 0, \varepsilon)_{c,d}$-tolerant tester will be called an $(r, s, \varepsilon)_{c,d}$-local tester. (ii) An $(r, s, 0, 1/n)_{c,d}$-tolerant tester will be called an $(r, s)_{c,d}$-error detector. There are some definitional issues that are resolved in the full version of this paper [17].

List Disjunct Matrices. We give a low-space algorithm that can compute a small set of possible defectives given an outcome vector which is generated by a list disjunct matrix [17].

Some Explicit Families of Codes. We now mention two explicit families of codes that we will see later on in the paper. We first begin with the Reed-Solomon code. Given $q \geq n \geq 1$ and a subset $S = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_q$, the Reed-Solomon code with *evaluation set* S and dimension k, denoted by $\mathrm{RS}_S[k]$, is defined as follows: Any message in $\mathbb{F}_q^k$ naturally defines a polynomial $P(X)$ of degree at most $k-1$ over $\mathbb{F}_q$. The codeword corresponding to the message is obtained by evaluating $P(X)$ over all elements in S. It is known that a $(n-k) \times n$ parity check matrix of RS_S is given by $H_{\mathrm{RS}_S} = \{v_j \cdot \alpha_j^i\}_{i=0, \, j=1}^{n-k-1, \, n}$, where $v_j = \frac{1}{\prod_{1 \leq \ell \leq n, \ell \neq j}(\alpha_j - \alpha_\ell)}$.

Another explicit code family we will consider are expander codes. These are binary codes whose parity check matrices are incidence matrices of constant-degree bipartite expanders. In particular, if we start with a strongly explicit expander, then the parity check matrix of the corresponding expander code will also be strongly explicit.

3 Data Stream Algorithms for Error-Detection

A positive result. We first show that any linear code with a strongly explicit parity check matrix has an efficient 1-pass data stream error detector.

Note that for a linear code $C \subseteq \mathbb{F}_q^n$ with parity check matrix H, the error detection problem with the usual polynomial time complexity setting is trivial. This is because by the definition of parity check matrix for any $\boldsymbol{y} \in \mathbb{F}_q^n$, $\boldsymbol{y} \in C$ if and only if $H \cdot \boldsymbol{y}^T = \mathbf{0}$. However, the naive implementation requires $\Omega(n)$ space which is prohibitive for data stream algorithms. We will show later that for deterministic data stream algorithms with a constant number of passes, this space requirement is unavoidable for asymptotically good codes.

However, the story is completely different for randomized algorithms. If we are given the *syndrome* $\boldsymbol{s} = H\boldsymbol{y}^T$ instead of $\boldsymbol{y}$ as the input, then we just have to solve the *set equality* problem which has a very well known one-pass $O(\log n)$-space data stream algorithm based on the *finger-printing* method. Because $\boldsymbol{s}$ is a fixed linear combination of $\boldsymbol{y}$ (as H is known), we can use the fingerprinting technique in our case. Further, unlike the usual fingerprinting method, which requires any large enough field, we need to use an extension field of $\mathbb{F}_q$. For this, we first need to get our hands on irreducible polynomials over $\mathbb{F}_q$, but fortunately this isn't difficult [17]. We now state our result.

Theorem 1. *Let $C \subseteq \mathbb{F}_q^n$ be a linear code of dimension k and block length n with parity check matrix $H = \{h_{i,j}\}_{i=0, \, j=1}^{n-k-1, n}$. Further, assume that any entry $h_{i,j}$ can be computed in space $\mathcal{S}(n,q)$, for some function $\mathcal{S}$. Given an $a \geq 1$, there exists a $(1, O(\mathcal{S}(n,q) + a \log n))_{1,n^{-a}}$-error detector for C.*

It is easy to check that $\mathcal{S}(q,n)$ is $O(\log n)$ for (strongly explicit) expander codes and RS codes. This implies the following:

Corollary 1. *Let $n \geq 1$. Then for $q = 2$ and $n \leq q \leq \text{poly}(n)$, there exists an asymptotically good code $C \subseteq \mathbb{F}_q^n$ that has a $(1, O(\log n))_{1,1/2}$-error detector.*

A negative result. We show that randomness is necessary even for local testing. In particular, we show the following [17]:

Theorem 2. *Let $C \subseteq [q]^n$ be a code of rate R and relative distance δ and let $0 \leq \varepsilon \leq \delta^2/8$ be a real number. Then any $(r, s, \varepsilon)_{1,0}$-local tester for C needs to satisfy $r \cdot s \geq \frac{\delta R n}{6}$.*

Error detection of a projected down code. We will be dealing with $\text{RS}_S[k]$ with $S = \{\alpha_1, \ldots, \alpha_n\}$. In particular, we are interested in a one-pass, low space data stream algorithm to solve the following problem: The input is $\boldsymbol{y} \in \mathbb{F}_q^n$ followed by a subset $E \subseteq S$ with $|E| = e$. We need to figure out if $\boldsymbol{y}_{S \setminus E} \in \text{RS}_{S \setminus E}[k]$. We have the following result [17]:

Lemma 1. *Let $e, n, k \geq 1$ be integers such that $k + e \leq n$. Then the problem above can be solved by a one-pass, $O(e + a \log n)$ space data stream algorithm with probability at least $1 - n^{-a}$, for any $a \geq 1$.*

4 Tolerant Testing

In this section we assume that we are working with $\text{RS}_S[k]$, where $S = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_q$. (However, our results will also hold for closely related codes such as the folded RS code [14].)

As was mentioned in the Introduction, there is a trivial reduction from one tolerant testing instance (say where we are interested in at most e vs. $> e$ errors) to $\binom{n}{e}$ instances of error detection: for each of the $\binom{n}{e}$ potential error locations, project the received word outside of those indices and check to see if it's a codeword in the corresponding RS code via the algorithm in Theorem 1. Using Theorem 1 (with $a = O(e)$), we can implement this as an $(1, O(e \log n), e/n, (e+1)/n)_{1, n^{-\Omega(e)}}$-tolerant tester. Unfortunately, this algorithm uses $\frac{n}{e}^{O(e)}$ time. Next, we show how to obtain roughly the same space complexity but with a much better time complexity.

Theorem 3. *Let $e, k, n \geq 1$ be integers such that $k \leq n$ and $e \leq n - k$. Then*

(a) If $e(e + k) \leq O(n)$, then there exists a $(1, O(e \log^2 n), e/n, 2e/n)_{1, n^{-\Omega(1)}}$-tolerant tester for $\text{RS}_S[k]$ under worst-case errors.
(b) If $ek \leq O(n)$, then there exists a $(1, O(e \log^2 n), e/n, 2e/n)_{1, n^{-\Omega(1)}}$-tolerant tester for $\text{RS}_S[k]$ under random errors.
(c) If $e \leq O(\sqrt[s+1]{sn/k})$, then there exists a $(1, O(e^2 \log^3 n), e/n, 5e/n)_{1, n^{-\Omega(1)}}$-tolerant tester for the folded RS code with folding parameter s under worst-case errors.

Further, all the algorithms can be implemented in $\tilde{O}(n^2)$ time.

In the above, the soundness parameter follows by picking a to be large enough while using Theorem 1. We observe that a naive implementation achieves the $\tilde{O}(n^2)$ run time. We also show that for part (a) and (b) by replacing n by $n/\log n$ in the RHS of the upper bound on k and bumping up the space to $O(e^2 \log^2 n)$, the algorithms for $\mathrm{RS}_{\mathbb{F}_q}[k]$ can be implemented in $\tilde{O}(ne)$ time. In fact, along with the faster running time, we get $(1, O(e \log^2 n), e/n, (e+1)/n)_{1,n^{-\Omega(1)}}$-tolerant testers [17]. For the rest of the section, we will focus on the other parameters of the algorithms.

All of the results above follow from a generic reduction that uses group testing. In particular, let $\boldsymbol{y}$ be the received word that we wish to test, and $\boldsymbol{c}$ be the nearest codeword to $\boldsymbol{y}$. Let $\boldsymbol{x} \in \{0,1\}^n$ be the characteristic vector associated with error locations in $\boldsymbol{y}$ with respect to $\boldsymbol{c}$. The high level idea is essentially to figure out $\boldsymbol{x}$ using group testing. We will need one definition and one proposition:

Definition 1. *Let $n, s_1, s_2, e, \ell, L \geq 1$ be integers with $s_1 \leq s_2$ and let $0 \leq \gamma \leq 1$ be a real. For any subset $T \subset [n]$ such that $|T| \leq e$, let $\mathcal{F}_{s_1,s_2}(T)$ be a collection of* forbidden *subsets of $[n]$ of size in the range $[s_1, s_2]$ such that $|\mathcal{F}_{s_1,s_2}(T)| \leq L$. A $t \times n$ binary matrix M is called a $(e, \ell, \gamma, \mathcal{F}_{s_1,s_2})$-*list disjunct *matrix if there exist integers $0 \leq b_1 < b_2$ such that the following hold for any $T \subseteq [n]$ with $|T| \leq e$:*

1. *For any subset $U \subseteq [n]$ such that $|U| \geq \ell$ and $U \cap T = \emptyset$, there exists an $i \in U$ with the following property: The number of rows where the ith column of M has a one and all the columns in T have a zero is at least b_2.*
2. *The following holds for at least $(1-\gamma)e$ many $i \in T$: Let R_i denote all the rows of M (thought of as subsets of $[n]$) that contain i. Then $|\{U \in R_i | U \subseteq V,$ for some $V \in \mathcal{F}_{s_1,s_2}(T)\}| \leq b_1$.*

This definition is a natural extension of (e, ℓ)-list disjunctness from [15], which is itself an extension of the (e)-disjunct matrix from [10].

Proposition 4. *Let $n, e, \ell, s_1, s_2, L, \gamma, \mathcal{F}_{s_1,s_2}$ be as in Definition 1. Let M be a $(e, \ell, \gamma, \mathcal{F}_{s_1,s_2})$- list disjunct matrix with t rows. Finally, consider an outcome vector $\boldsymbol{r}$ of applying M to a set of defectives E with $|E| \leq e$ in the $(e, \mathcal{F}_{s_1,s_2})$-group testing scenario. Then there exists an algorithm $\mathcal{A}$, which given $\boldsymbol{r}$ can compute a set G such that $|G| \leq \ell + e - 1$ and $|E \setminus G| \leq \gamma e$. Further, $\mathcal{A}$ uses $O(t + \log n + S(t,n))$ space, where $S(t,n)$ is the space required to compute any entry of M.*

Let M be a $t \times n$ binary matrix that is (e,e)-list disjunct. We can get our hands on M with $t = O(e \log n)$ with $O(e \log^2 n)$ space [17]. Now consider the following natural algorithm:

> For all $i \in [t]$, check if $\boldsymbol{y}_{M_i} \in RS_{M_i}[k]$, where M_i is the subset corresponding to the ith row of M. If so, set $r_i = 0$, else set $r_i = 1$. Run $\mathcal{A}$ from Proposition 4 with $\boldsymbol{r}$ as input, to get $\hat{\boldsymbol{x}}$. (Step 1) If $\mathrm{wt}(\hat{\boldsymbol{x}}) \geq 2e$, declare that $\geq 2e$ errors have occurred. (Step 2) If not, declare $\leq e$ errors iff $\boldsymbol{y}_{S \setminus T} \in \mathrm{RS}_{S \setminus T}[k]$, where T is the subset corresponding to $\hat{\boldsymbol{x}}$.

The way the algorithm is stated above, it seems to require two passes. However, using Lemma 1, we can run STEP 2 in parallel with the rest of the algorithm, resulting in a one-pass implementation.

Let $\boldsymbol{z}$ be the result of applying M on $\boldsymbol{x}$. Now if it is the case that $z_i = 1$ iff $r_i = 1$, then the correctness of the algorithm above follows from the fact that M is (e, e)-list disjunct and Proposition 4. (If $\text{WT}(\boldsymbol{x}) \leq e$, then we have $S_{\boldsymbol{x}} \subseteq S_{\hat{\boldsymbol{x}}}$ (where $S_{\boldsymbol{x}}$ is the subset of $[n]$ whose incidence vector is $\boldsymbol{x}$) and $\text{WT}(\hat{\boldsymbol{x}}) < 2e$, in which case the algorithm will declare at most e errors. Otherwise, the algorithm will "catch" the at least e errors in either STEP 1 or failing which, in STEP 2.)

However, what complicates the analysis is the fact that even though $z_i = 0$ implies $r_i = 0$, the other direction is not true. In particular, we could have $\boldsymbol{y}_{M_i} \in \text{RS}_{M_i}[k]$, even though $(\boldsymbol{y} - \boldsymbol{c})_{M_i} \neq \boldsymbol{0}$. The three parts of Theorem 3 follow from different ways of resolving this problem.

Note that if $\text{WT}(\boldsymbol{x}) \geq 2e$, then we will always catch it in STEP 2 in the worst-case. So from now on, we will assume that $0 < \text{WT}(\boldsymbol{x}) \leq e$. Let the minimum support of any row in M be s.

We begin with part (a). Let $s > k+e$ and define $\Delta = \boldsymbol{y} - \boldsymbol{c}$. Note that we are in the case where $0 < \text{WT}(\Delta) \leq e$. Since $s \geq k$ and $\boldsymbol{c}_{M_i} \in RS_{M_i}[k]$, $\boldsymbol{y}_{M_i} \in RS_{M_i}[k]$ if and only if $\Delta_{M_i} \in RS_{M_i}[k]$. Note also that for any i, $\text{WT}(\Delta_{M_i}) \leq \text{WT}(\Delta) \leq e$. Now, the distance of $RS_{M_i}[k]$ is $s - k - 1 > e$, so for every i with non-zero Δ_{M_i}, $\Delta_{M_i} \notin \text{RS}_{M_i}[k]$, which in turn means that $z_i = 1$ will always imply that $r_i = 1$ when M has the stated support. It can be shown that $s \geq n/(2e)$ [17], which concludes the proof of part (a).

The following lemma follows from the random errors result in [18] and is needed for part (b):

Lemma 2 ([18]). *Let $k \leq n < q$ be integers such that $q > \left(\frac{n}{k}\right)^2$. Then the following property holds for RS codes of dimension k and block length n over $\mathbb{F}_q$: For $\geq 1 - q^{-\Omega(k)}$ fraction of error patterns $\boldsymbol{e}$ with $\text{WT}(\boldsymbol{e}) \leq n - 4k$ and any codeword $\boldsymbol{c}$, the only codeword that agrees in $\geq 4k$ positions with $\boldsymbol{c} + \boldsymbol{e}$ is $\boldsymbol{c}$.*

Now if $s \geq 4k$, then with high probability, every non-zero $\Delta_{M_i} \notin \text{RS}_{M_i}[k]$ (where Δ is as defined in the proof of part (a)). The fact that $s \geq n/(2e)$ completes the proof of part (b).

The proof of part (c) is more involved and needs a strong connection to the list decodability of the code being tested, which we discuss next.

Connection to List Decoding. Unlike the proofs of part (a) and (b) where the plain vanilla (e, e)-list disjunct matrix works, for part (c), we need and use a stronger notion of list disjunct matrices. We show that if the list disjunct matrix is picked at random, the bad tests (i.e. $r_i = 0$ even though $z_i = 1$) do not happen often and thus, one can decode the result vector even with these errors. We show that these kind of matrices suffice as long as the code being tested has good enough list decodability. The tolerant testing algorithm for a Reed-Solomon code, for instance, recursively reduces the amount of errors that need to be detected, and after application of Lemma 1, can be made to accomplish this in a single pass. We also show that list disjunct matrices with relevant parameters

from Definition 1 can be found, with high probability, using low space and a low number of random bits [17].

The space requirement of $O(e^2 \log^2 n)$ of part (c) is unsatisfactory. Reducing the amount of randomness needed to something like $O(e \log n)$ will realize the full potential our algorithm. We leave this as an open problem.

5 Limitations of Our Techniques

One shortcoming of Theorem 3 is that to distinguish between (say) at most e and at least $2e$ errors, we needed $e \cdot s \leq O(n)$, where s is the minimum support size of any test. Another shortcoming is that we need $O(e \log n)$ space. In this section, we show that our techniques cannot overcome these limits.

We begin with some quick notation. For any $k \geq 1$, a k^{++} query to a string $x \in \{0,1\}^n$ corresponds to a subset $S \subseteq [n]$. The answer to the query is x_S if $\text{WT}(x_S) < k$, otherwise the answer is k^{++} (signifying that $\text{WT}(x_S) \geq k$). (This is a natural generalization of k^+ decision trees considered by Aspnes et al. [3].) A k^{++} algorithm to solve the (ℓ, t, n)-threshold function makes a sequence of k^{++} queries to the input $x \in \{0,1\}^n$, and can tell whether $\text{WT}(x) \leq \ell$ or $\text{WT}(x) \geq t$. If we think of x as being the indicator vector for error locations, then our reduction from tolerant testing to error detection can be thought of as a 1^{++} algorithm for the $(e, O(e))$-threshold function.

First we show that the minimum support size that we obtain in our reduction, even with the stronger k^{++} primitive, is nearly optimal [17].

Theorem 5. *Let $0 \leq \ell < t \leq n$ and $k \geq 1$ be integers. Let $\varepsilon < 1/2$ be a constant real. Then any non-adaptive, randomized k^{++} algorithm for the (ℓ, t, n)-threshold problem with error probability at most ε, where all the queries have support size at least s, needs to make at least $e^{s\ell/n}/n^{O(k)}$ queries. In particular, any algorithm that makes a sublinear number of queries needs to satisfy $s \cdot \ell \leq O(kn \log n)$.*

Note that our reduction maps one tolerant testing problem instance (where say we want to distinguish between at most e error vs. at least $2e$ errors) to $O(e \log n)$ many instances of error detection. Next we show that this is essentially unavoidable even if we use k^{++} queries for constant k. The following result follows from the results in [3] and is proven in the full version of the paper [17].

Theorem 6. *Let $0 \leq \ell < t \leq n$ and $k \geq 1$ be integers. Then any* adaptive*, deterministic k^{++} algorithm for the (ℓ, t, n)-threshold problem makes $\Omega(\ell/k)$ queries.*

Acknowledgments

We thank Venkat Guruswami, Steve Li and Ram Swaminathan for helpful discussions. Thanks to Chris Umans for pointing out [4] to us.

References

1. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM 45(3), 501–555 (1998)
2. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. J. ACM 45(1), 70–122 (1998)
3. Aspnes, J., Blais, E., Demirbas, M., O'Donnell, R., Rudra, A., Uurtamo, S.: k+ decision trees (2010) (manuscript)
4. Bellare, M., Rompel, J.: Randomness-efficient oblivious sampling. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS), pp. 276–287 (1994)
5. Ben-Sasson, E., Guruswami, V., Kaufman, T., Sudan, M., Viderman, M.: Locally testable codes require redundant testers. In: IEEE Conference on Computational Complexity, pp. 52–61 (2009)
6. Ben-Sasson, E., Harsha, P., Raskhodnikova, S.: Some 3cnf properties are hard to test. SIAM J. Comput. 35(1), 1–21 (2005)
7. Chen, C.L., Hsiao, M.Y.: Error-correcting codes for semiconductor memory applications: A state-of-the-art review. IBM Journal of Research and Development 28(2), 124–134 (1984)
8. Chen, P.M., Lee, E.K., Gibson, G.A., Katz, R.H., Patterson, D.A.: RAID: High-performance, reliable secondary storage. ACM Computing Surveys 26(2), 145–185 (1994)
9. Dinur, I.: The PCP theorem by gap amplification. J. ACM 54(3), 12 (2007)
10. Du, D.-Z., Hwang, F.K.: Combinatorial Group Testing and its Applications. World Scientific, Singapore (2000)
11. Elerath, J.: Hard-disk drives: The good, the bad, and the ugly. Communications of the ACM 52(6), 38–45 (2009)
12. Füredi, Z.: On r-cover-free families. J. Comb. Theory, Ser. A 73(1), 172–173 (1996)
13. Guruswami, V., Rudra, A.: Tolerant locally testable codes. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX 2005 and RANDOM 2005. LNCS, vol. 3624, pp. 306–317. Springer, Heidelberg (2005)
14. Guruswami, V., Rudra, A.: Explicit codes achieving list decoding capacity: Error-correction up to the Singleton bound. IEEE Transactions on Information Theory 54(1), 135–150 (2008)
15. Indyk, P., Ngo, H.Q., Rudra, A.: Efficiently decodable non-adaptive group testing. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1126–1142 (2010)
16. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1997)
17. Rudra, A., Uurtamo, S.: Data stream algorithms for codeword testing. arXiv:1004.4601 [cs.IT] (2010)
18. Rudra, A., Uurtamo, S.: Two theorems in list decoding. ECCC Technical Report TR10-007 (2010)
19. Spielman, D.: Linear-time encodable and decodable error-correcting codes. IEEE Transactions on Information Theory 42(6), 1723–1732 (1996)

Streaming Algorithms for Independent Sets

Bjarni V. Halldórsson[1], Magnús M. Halldórsson[2,⋆],
Elena Losievskaja[2], and Mario Szegedy[3,⋆⋆]

[1] School of Science and Engineering, Reykjavik University, 101 Reykjavik, Iceland
bjarnivh@ru.is
[2] School of Computer Science, Reykjavik University
mmh@ru.is, ellossie@gmail.com
[3] Dept. of Computer Science, Rutgers University, 110 Frelinghuysen Road,
Piscataway, New Jersey, USA
szegedy@cs.rutgers.edu

Abstract. We find "combinatorially optimal" (guaranteed by the degree-sequence alone) independent sets for graphs and hypergraps in linear space in the semi-streaming model.

We also propose a new *output-efficient* streaming model, that is more restrictive than semi-streaming ($n \cdot \log^{O(1)} n$ space) but more flexible than classic streaming ($\log^{O(1)} n$ space). The algorithms in this model work in poly-logarithmic space, like in the case of the classical streaming model, but they can access and update the output buffer, treating it as an extra piece of memory.

Our results form the first treatment of the classic IS problem in the streaming setting.

1 Introduction

In this paper we consider streaming algorithms for the classic independent set problem on graphs and hypergraphs. As input, we are presented with a (hyper)graph edge by edge, and we have to output a large set of vertices that contains no (full) edges. For graphs, a theorem of Paul Turan guarantees an independent set of size $n/(d+1)$, where d is the average degree of the graph. If the entire degree-sequence, $d_1, \ldots, d_n$ of the graph is available, then $\sum_{i=1}^{n} \frac{1}{d_i+1}$ is a stronger lower bound for the maximum independent set size (in this paper n will always denote the number of nodes of the input graph). The formula has a generalization for hypergraphs as well (see Section 1.2). This "combinatorially optimal" output size is what we will require of our algorithms.

In the *streaming model* [13], the data is presented sequentially in the form of a data stream, one item at a time, and the working space (the memory used by the algorithm) is significantly less than the size of the data stream. The motivation for the streaming model comes from practical applications of managing massive data sets such as, e.g., real-time network traffic, on-line auctions, and telephone call records. These data sets are huge and arrive at very high rate, making it

⋆ Supported by grant 7000921 from the Iceland Research Fund.
⋆⋆ Supported by NSF grant EMT-0523866.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 641–652, 2010.

impossible to store the entire input and forcing quick decisions on each data item. However, most graph problems are impossible to solve within the polylogarithmic space bound that the traditional streaming model offers (a rare exception is an algorithm for the problem of counting triangles in a graph [5]).

This observation led to the introduction of the *semi-streaming* model [13], where space for n-vertex graphs is restricted to $n \log^{O(1)} n$. The algorithms then have enough space for some information for each vertex, but not enough to store the whole graph. A number of graph problems have been studied in the semi-streaming model, including bipartite matching (weighted and unweighted cases) [10,9], diameter and shortest paths [10,11], min-cut [1], and graph spanners [11]. The independent set (IS) problem, to our best knowledge, has not been studied in the model before.

In the case of IS, the $n \log^{O(1)} n$-bound of semi-streaming seems overly generous, but we cannot reasonably expect a sublinear bound, given the need to store the problem solution. Instead, we suggest that the focus be placed on the amount of *extra space* required, i.e. in addition to that required for storing the solution.

All the algorithms we consider have the common feature that they maintain a feasible solution at all times. Also, decisions made on edges are irreversible: once a node is ejected from the solution, it never enters it again. This defines a new *online streaming* model. It is closely related to *preemptive online* algorithms, considered recently by Epstein et al.[9] in a streaming context for weighted matching. The difference is that in our problem, we can view the whole vertex set as belonging to the initial solution, thus the solution of any algorithm is *monotonously non-increasing* with time. There have been few works on online graph problems that allow for one-way changes. A rare example is a recent work of [8] that can be viewed as dealing with maintaining a strong independent set of a hypergraph when edges arrive in a stream.

To contrast, in the classical *online* version of the IS problem [12,2], vertices arrive one by one, along with all incident edges to previous vertices. The online algorithm must then determine once and for all whether the node is to be included in the constructed feasible independent set solution. This problem is very hard to approximate [12,2]; e.g., a competitive factor of $n-1$ is best possible for deterministic algorithm, even when restricted to trees. However, bounded-degree graphs are comparatively easy, since a factor of Δ is trivial for a deterministic greedy algorithm.

By focusing on the space requirements and the way the working space can be used, we seek to gain a deeper understanding of the way independent sets can be computed. We generally try to avoid any prior assumptions such as knowing the set or the number of vertices in advance, and we are willing to pay a well-calculated price for this in terms of extra space.

1.1 Our Contribution

We design space-efficient semi-streaming algorithms for IS. Our starting point is algorithm RandomOffLine of Shachnai and Srinivasan [14] (see Fig. 2). We modify this algorithm to achieve:

- $O(n \log r)$ space, instead of $n \log n$, where r is the cardinality of the largest hyperedge (Algorithm RandomPartialPermute). Also, n does not need to be known for the algorithm in advance.
- $O(n)$ space (Algorithm RandomMap; here n needs to be known in advance, and the constants are somewhat compromised).

We also analyse the situation, where we are allowed to keep only a single bit (!) per node.

Finally, we explore new models of semi-streaming and argue upper and lower bounds. In the on-line semi-streaming model output-changes can go only in one direction and a feasible solution must be maintained at all times. Also, *memoryless* algorithms are allowed only logarithmic *extra* space in addition to the bits required to store the solution.

1.2 Definitions

Given a hypergraph $H = (V, E)$, let n and m be the number of vertices and edges in H, respectively. We assume that H is a *simple* hypergraph, i.e. no edge is a proper subset of another edge. An *independent set* I in H is a subset of vertices that contains no edge of H. If I is independent, then $V \setminus I$ is said to be a *hitting set*.

A hypergraph is r-*uniform* if all edges have the same cardinality r. Graphs are exactly the 2-uniform hypergraphs. For graphs and hypergraphs the *degree* $d(v)$ of a vertex v is the number of edges incident on v. We denote by Δ and d the maximum and the average degree, respectively. For us, a much better measure for the degree of the vertex v of H is $1/p$, where p is the solution of $\sum_{e \ni v} p^{|e|-1} = 1$. This we call the *efficient degree* of v, and we denote it by $d^*(v)$. Also, define $i(H) = \sum_v \frac{1}{d^*(v)}$. For intuition, note that for a d-regular k-uniform hypergraph H, $i(H) = n/\sqrt[k-1]{d}$. Let $\alpha(H)$ be the maximum independent set size of H. What makes $i(H)$ interesting for us is that $\alpha(H) = \Omega(i(H))$. In fact, $\Omega(i(H))$ is the strongest lower bound for $\alpha(H)$ we can obtain from the degree-sequence alone. Let IS denote the problem of finding an independent set in hypergraphs of size $\Omega(i(H))$.

We can generalize all of the definitions for weighted (hyper)graphs. A vertex-weighted hypergraph has a non-negative weight function on its vertices. The notions of average degree etc. carry over to the weighted case in a natural way, for instance $i(H)$ becomes $\sum_v \frac{w(v)}{d^*(v)}$, where w is the weight function. Most of our results will carry over to the weighted case with obvious modifications.

We also note that our algorithms will be such that if vertices with degree 0 should appear in H, then they will be automatically included in the independent set that the algorithm outputs.

We assume that the vertices are labelled $0, \ldots, n-1$. This assumption can be voided by simply maintaining a lookup table, but the storage requirements for such lookup are beyond the scope of our considerations here.

For this article, the most precious resource is space, and we model and regard memory as a linear array of bits with direct access (as in standard C programming).

2 A Basic Algorithm

All of our algorithms will be based on the well known observation, that in a sparse graph random sets of the right size are nearly independent. The algorithm we present in Fig. 1 is crucial to the analysis of the subsequent algorithms in later sections.

Remark. On a random set of size pn we shall mean either of the two things: 1. We select a subset of $V = [n]$ of size pn uniformly and randomly. 2. We create a random subset X of V by the procedure that selects all nodes in V randomly and independently with probability p. While we prove Lemma 1 only for case 2, our applications will use lemma for case 1, where it also holds.

ALGORITHM BASIC[p, H]
Input: a hypergraph $H(V, E)$, probability value p

$S \leftarrow \emptyset$
Let X be a random subset of V of size pn.
For each edge $e \in E(H)$ do
    If $e \not\subseteq X$, let v be any vertex in $e \setminus X$;
    otherwise, let v be any vertex of e.
    $S \leftarrow S \cup \{v\}$
Output $I = V \setminus S$

Fig. 1. Algorithm BASIC is crucial for most of our performance analysis

We now analyze BASIC[p, H].

Lemma 1. *Let $H(V, E)$ be a hypergraph and let A be the set of nodes of H with efficient degree $\leq \frac{1}{2p}$. Then* BASIC[p, H] *will return an independent set I such that $\mathbb{E}[|A \cap I|] \geq p|A|/2$.*

Proof. For a node v, let χ_v be the random variable that takes value 1 if the node is selected into I, and 0 otherwise. Then $|A \cap I| = \sum_{v \in A} \chi_v$. To estimate the expectation of χ_v from below, notice that if v is in X, but for every edge e incident on v we have $e \not\subseteq X$, then $v \in I$. The probability that $v \in X$ is p. The conditional probability that for an e with $v \in e$ we have that $e \not\subseteq X$ is $1 - p^{|e|-1}$. If the edges incident on v would have only vertex v in common, and were otherwise disjoint, then this would give us the probability

$$p \prod_{e:\, v \in e} \left(1 - p^{|e|-1}\right) \geq p \left(1 - \sum_{e:\, v \in e} p^{|e|-1}\right) \geq p/2$$

for the desired event, where the last inequality follows from $\sum_{e:\, v \in e} (2p)^{|e|-1} \leq 1$, since the efficient degree of any $v \in A$ by our assumption is at least $1/(2p)$. When the (hyper)edges that are incident on v arbitrarily intersect, we use the FKG-inequality as in [14] towards getting the exact same estimate. The latter implies that the correlations that arise by letting the edges overlap will make the event that none of them is contained in X more likely (we omit the details).

The lemma now follows from the additivity of expectation.

3 Permutation-Based Algorithms

The idea of permutation-based algorithms is to randomly permute vertices and use this random permutation to decide which vertices to include in the independent set. Given a set of vertices V, the randomized algorithm RandomOffLine (see Fig. 2) creates a permutation π on the given set V of vertices. The last vertex of each edge is added to a set S. The set S forms a hitting set and $V \backslash S$ an independent set.

Algorithm RandomOffLine
Input: a hypergraph $H(V, E)$

$S \leftarrow \emptyset$
Let π be a random permutation of the vertices in V
For each edge $e \in E(H)$ do
    Let v be the last vertex in e with respect to π
    $S \leftarrow S \cup \{v\}$
Output $I = V \backslash S$

Fig. 2. The off-line algorithm RandomOffLine

Shachnai and Srinivasan [14] proved the following performance bounds for RandomOffLine on hypergraphs. An elegant proof for graphs is featured in the book of Alon and Spencer [4].

Theorem 1 ([14]). *Given a hypergraph H, the off-line algorithm* RandomOffLine *finds an independent set of expected weight $\Omega(i(H))$.*

A simple heuristic improvement to the algorithm is to add the last vertex of an edge to S only if that edge doesn't already have a representative in the set, i.e., if $e \cap S \neq \emptyset$. Since the addition of this condition never decreases the solution size, the performance claims continue to hold.

It is immediately clear that RandomOffLine is actually a streaming algorithm, since it treats the edges in whichever given order. We can even avoid the assumption that the algorithm knows the number n of vertices in advance. For that, we construct a random permutation π on-the-fly by randomly inserting each new vertex in the ordering of the previous vertices.

We can significantly reduce the space by constructing a partial random permutation instead of a complete random permutation. Each vertex v is associated with a finite sequence s_v of random bits, where each bit in s_v is independently set to 0 or 1 with equal probability. A partial random permutation of the vertex set V is then a lexicographic order of the corresponding bit-sequences $\{s_v\}_{v \in V}$. Consider an edge $e \in E$. Let $\{s_v | v \in e\}$ be the set of bit-sequences associated with the vertices in e. For vertices $u, v \in e$, we write $u \succ v$ if s_u follows s_v in lexicographical order. We define the vertex $u \in e$ to be the *last* in e, if s_u is lexicographically last in the set $\{s_v | v \in e\}$. The idea is that for each vertex $v \in e$ we use the minimum number of bits required to determine if s_v is the last in $\{s_v | v \in e\}$. In other words, we just need to determine which vertex in e is the last, and the relative order of other vertices in e is not important. The formal description of this algorithm RandomPartialPermute is given in Fig. 3. Let $s_v[j]$ be the j-th bit in s_v.

Algorithm RandomPartialPermute
Input: a stream E of edges

$V \leftarrow S \leftarrow \emptyset$
For each edge e in the stream E do
  For each vertex $v \in e$ such that $v \notin V$ do
    $V \leftarrow V \cup \{v\}$
    $s_v \leftarrow \emptyset$
  $U \leftarrow \{e\}$
  $j \leftarrow 1$
  While $|U| \neq 1$ do
    For each $v \in U$ such that $s_v[j]$ is not defined do
      $s_v[j] = 1$ with probability $\frac{1}{2}$, otherwise $s_v[j] = 0$
    If $\exists v \in U$ such that $s_v[j] = 1$
      $U \leftarrow U \backslash \{v \in U \mid s_v[j] = 0\}$
    $j \leftarrow j + 1$
  $S \leftarrow S \cup U$
Output $I = V \backslash S$

Fig. 3. The algorithm RandomPartialPermute

As stated, the algorithm RandomPartialPermute is not fully implemented. Specifically, it remains to organize the bits stored into a structure that can be easily accessed. Various approaches are possible that all complicate the picture. We will instead leave the idea in this partially developed state.

Theorem 2. RandomPartialPermute *finds an independent set of expected weight* $\Omega(i(H))$ *using expected* $O(n \log r)$ *space and* $O(r)$ *time to process each edge.*

Proof. First, we show that the ordered set $S = \{s_v | v \in V\}$ forms a partial permutation of V. We note that a random permutation can be created by assigning each

vertex a random number drawn from the uniform distribution on $[0, 1)$ and asserting that u follows v in π, if the random number assigned to u is greater than the random number assigned to v. A uniform random number drawn from $[0, 1)$ can be viewed as an infinite sequence of unbiased random bits. RandomPartialPermute creates such a bit-sequence with only as many bits created as needed to determine if s_v is lexicographically last in $\{s_v | v \in e\}$ for every edge e such that $v \in e$. Therefore, the set $\{s_v | v \in V\}$ forms a partial permutation of V and we can apply the same argument as in the proof of Theorem 1 to show that RandomPartialPermute finds an independent set of expected weight $\Omega(i(H))$.

In the remainder we show that the algorithm uses $O(n \log r)$ space to store the set $\{s_v\}$ of bit-sequences. Given a vertex v, we say that we *open* the j-th bit in s_v, if the algorithm assigns $s_v[j]$ to either 0 or 1.

Consider an edge e incident on v. Let $s_{v,e}$ be the bit-sequence s_v at the time e appears in the stream and let $u \succ v$ if $s_{u,e} > s_{v,e}$. Let $U_{\succ v}(e) = \{u \in e | u \succ v\}$, $U_{\prec v}(e) = \{u \in e | v \succ u\}$ and $U_{=v}(e) = \{e\} \setminus (U_{\succ v}(e) \cup U_{\prec v}(e))$. We need to open more bits in $s_{v,e}$ only if $U_{\succ v}(e) = \emptyset$ and $U_{=v}(e) \neq \emptyset$, because in this case the vertices in $U_{=v}(e)$ have the highest bit-sequences and these bit-sequences are exactly the same as $s_{v,e}$. In this case we say that e is *problematic for* v. In any other case, the opened bits in $s_{v,e}$ are sufficient to decide which vertex covers e, namely e is covered either by a vertex $u \in U_{\succ v}(e)$ if $U_{\succ v}(e) \neq \emptyset$ or by the vertex v if $U_{\succ v}(e) = \emptyset$ and $U_{=v}(e) = \emptyset$.

To simplify the analysis we will open bits in batches, $3 \log r$ bits are opened initially and immediately following the resolution of a problematic edge, and further as many bits are opened as are needed to resolve a problematic edge.

Consider an edge e problematic for v. We compute an upper bound on the expected number of vertices that have the same bit-sequence as v and condition the computation of this expectation over all the values that v can take. Each time we encounter a problematic edge we are guarenteed to have $3 \log r$ bits which may be considered to be random. The bit-sequences that the other vertices in e take are conditioned on being less than or equal to the bit-sequence for v. Then, the expected number of vertices that have the same bit-sequence as v in e at the $3 \log r$ bits under consideration can be determined by: summing over all r^3 possible values that the bit-sequence for v can take and multiplying the probability that v takes the value of a given bit-sequence, $\frac{1}{r^3}$, with the expected number of other nodes in e that take the same bit-sequence value. The expected number of nodes in e taking the same bit-sequence value as v is bounded by $\frac{(r-1)}{i+1}$, where i is the bit-sequence value of v (As $| e - \{v\} | \leq r - 1$ and $i + 1$ is the number of bit-sequences $\leq i$). We get an expression

$$|U_{=v}(e)| \leq \sum_{i=0}^{r^3} \frac{1}{r^3}(r-1)\frac{1}{i+1} \leq \frac{(3 \log r + 1)\frac{r-1}{r}}{r^2} \leq \frac{1}{r} \ .$$

Each time we encounter a problematic edge e, we open $3 \log r$ bits and then as many bits as needed to determine which of the vertices in $U_{=v}(e)$ has the highest bit-sequence, in expectation $\log(1 + \frac{1}{r})$ bits. In total we open in expectation

$$\sum_{i=0}^{\infty} \left(3\log r + \log(1+\frac{1}{r})\right) \left(\frac{1}{r}\right)^i \leq 6\log r$$

bits.

3.1 Linear Space Algorithm

Interestingly, we can run algorithm Basic "in parallel," for many different p's at the same time in the same *sequential* algorithm! This happens in the case of algorithm RandomOffLine. For a random permutation π, define

$$X_k = \{\text{first } k \text{ elements of } \pi\} \ .$$

Now X_k is a random set with size pn, where $p = k/n$. Notice that upon running RandomOffLine we also run Basic for X_k for $k = 1, \ldots, n$ in parallel. Indeed, it holds that when a hyperedge e is processed, we add the last vertex, v, of e to S. Thus, unless $e \subseteq X_k$, vertex v is not in X_k. Using this property, we see that for every vertex v the expectation of χ_v of Lemma 1 is at least $\frac{1}{4(d^*(v)+1)}$, just by picking X_k for the node with $k = n/d^*$, and applying the same argument to bound $\mathbb{E}[\chi_v]$ from below, as in Lemma 1. Theorem 1 (aside from a constant factor) now follows easily from the additivity of expectation.

We now create a new, more efficient algorithm, RandomMap from the observation that the above argument goes through even when we only use the properties of X_k for $k = 1, 2, 4, 8, \ldots$. Indeed, if v has efficient degree d^* then choose $\frac{n}{2d^*} \leq k \leq \frac{n}{d^*}$. Then, we get that χ_v of Lemma 1 is at least $\frac{1}{8(d^*(v)+1)}$.

Algorithm RandomMap
Input: a hypergraph $H(V, E)$

$S \leftarrow \emptyset$
Let ρ be a random map $V \rightarrow [\log n]$ described below.
For each edge $e \in E(H)$ do
  Let v be the vertex in e with the largest image in ρ
    (with conflicts resolved arbitrarily)
  $S \leftarrow S \cup \{v\}$
Output $I = V \backslash S$

Fig. 4. Algorithm RandomMap

To provide the X_k, for all powers of two, we do not need to create a permutation of the nodes. A map $\rho : V \rightarrow [\log n]$ suffices, where the probability of $\rho(v) = i$ is $1/2^i$ (to make the total probability equal to one, we set the probability of $\rho(v) = \log n$ to be $2/n$ instead of $1/n$).

Our new algorithm runs in the same way as RandomOffLine, except that π is replaced by ρ, and when conflicts (i.e., when the smallest element is not unique) are resolved arbitrarily.

We now describe how to store and access ρ using small space. Since the domain of ρ has size $\log n$, there is a straightforward random access implementation that uses only space $n \log\log n$ by storing $\rho(i)$ in the memory segment $[(i-1)\lceil\log\log n\rceil+1, i\lceil\log\log n\rceil]$. We can use the space even more efficiently, however. Notice that $\sum_{i=1}^{n} \log \rho(i) = O(n)$. Thus, the sequence

$$\rho(1)\ \$\ \rho(2)\ \$ \ldots \$\ \rho(n)$$

has bit-length linear in n. But this does not facilitate the binary search for $\rho(i)$, since there is no indication in the above sequence which ρ value we are reading. And since the $\rho(i)$'s have random bit length, we cannot directly access the address of $\rho(i)$ either. One of several possible solutions is to insert markers in $n/\log n$ places to guide our search. We put a marker before $\rho(1)$, $\rho(1+\log n)$, $\rho(1+2\log n)$, and so on. Each marker has bit-length $\log n$, and tells which ρ-value comes immediately after it. The new sequence with the markers obviously still occupies linear space. It is easy to see that now we can implement a binary search where each step of the search involves finding the closest marker and to decide whether to go left or right. Then after each step the search interval is halved (in terms of bit-length). At the final step a short sequential search leads to the desired $\rho(i)$. The worst case running time is $O((\log n)^3)$. There are various ways to make this algorithm more time-efficient.

4 Online Streaming Algorithms

All the algorithms considered in this paper have the following two properties: (a) they maintain a feasible solution I at all times, and (b) rejection decisions (i.e., the removal of a node from I, or alternatively addition to S) are irrevocable. We refer to such algorithms as *online streaming* algorithms.

In this section, we study the power of this model in the deterministic case. We restrict our attention here to the case of graphs.

Deterministic algorithms: The next result shows that no deterministic algorithm can attain a performance ratio in terms of d alone, nor a ratio of $2^{o(\Delta)}$.

Theorem 3. *The performance ratio of any deterministic algorithm in the online streaming model is $\Omega(n)$. This holds even for trees of maximum degree $\log n$, giving a bound of $\Omega(2^{\Delta})$. It also holds even if the algorithm is allowed to use arbitrary extra space.*

Proof. Assume that $n = 2^k$ is a power of 2. Let A be any deterministic algorithm.

We maintain the invariant that the independent set selected by A contains at most one node in each connected component. We join the n vertices together into a single tree in k rounds. In round i, for $i = 1, 2, \ldots, k$, $n/2^i$ edges are presented. Each edge connects together two components; in either component, we choose as endpoint the node that is currently in A's solution, if there is one, and otherwise use any node in the component. This ensures that the algorithm cannot keep both vertices in its solution, maintaining the invariant.

In the end, the resulting graph is a tree of maximum degree at most k, and A's solution contains at most one node.

When allowing additional space, we can match the previous lower bound in terms of Δ. The proof is omitted for lack of space.

Theorem 4. *There is a deterministic algorithm in the online streaming model with a performance ratio of* $O(2^{\Delta})$.

5 Minimal Space Algorithms

In the most restricted case, we have no extra space available. We can refer to such algorithm as *memoryless*, since they cannot store anything about previous events. Can we still obtain reasonable approximations to IS?

We show that there exists a function g allowing us to find a $n/g(d)$-independent set in this model, but that g must now be exponential.

The algorithm RandomDelete given in Fig. 5 selects an endpoint at random from each edge in the stream and removes it from the current solution. Note that RandomDelete is memoryless. For the sake of simplicity, we restrict our attention in this section to the case of graphs.

Algorithm RandomDelete
Input: a stream E of edges

$V \leftarrow S \leftarrow \emptyset$
For each edge e in the stream E do
  $V \leftarrow V \cup e$
  Randomly select a vertex $v \in e$
  $S \leftarrow S \cup \{v\}$
Output $V \setminus S$

Fig. 5. The algorithm RandomDelete

Intuitively, a memoryless algorithm would seem to be unable to do significantly better than randomly selecting the vertex to be eliminated.

Theorem 5. RandomDelete *finds an independent set of expected weight* $n/2^{O(d)}$*, and this is tight even if the algorithm avoids eliminating vertices unnecessarily.*

Proof. Upper bound. Each vertex v belongs to the final solution $V \setminus S$ with probability $2^{-d(v)}$. Therefore, the expected size of the $V \setminus S$ is $\sum_{v \in V} 2^{-d(v)} \geq n/2^d$, using the linearity of expectation and Jensen's inequality.

Lower bound. Consider the graph with vertex set $V = \{v_1, v_2, \cdots, v_n\}$ and edges $\{v_i, v_j\}$ for any $|i - j| \leq k$. Edges arrive in the stream in lexicographic order: (v_1, v_2), (v_1, v_3), $\ldots, (v_1, v_k)$, $(v_2, v_3), \ldots, (v_{k+1})$, (v_3, v_4), etc. Note, that

all but the first and the last k vertices have degree $2k$. Thus, the average degree $d \leq \Delta = 2k$.

Let I be the independent set constructed by the algorithm. Consider the first vertex v_1. There are two cases, depending on whether v_1 ends up in I.

Case 1: $v_1 \in I$. It means that all the neighbors of v_1 are deleted. The probability of this event is $P[v_1 \in I] = 2^{-k}$. The remaining stream is identical to the original one with $V = V \setminus \{v_1, v_2, \cdots, v_k\}$.

Case 2: $v_1 \notin I$. Suppose v_1 was selected to cover the t-th edge incident on v_1 for some $t \in [1, k]$. Then, the first $t-1$ neighbors of v_1 were selected to cover the first $t-1$ edges incident on v_1 and were deleted as well. The remaining stream is identical to the original one with $V = V \setminus \{v_1, v_2, \cdots, v_t\}$.

Thus, a vertex $v_i \in V$ is inserted in I only in Case 1 and the probability of this event is 2^{-k}, for any $i \in [1, n-k]$. Note, that the last k vertices form a clique, and so only one vertex from this clique contributes to I. Thus, the expected size of the independent set found by the algorithm is at most $\frac{n-k}{2^k} + 1 < \frac{n}{2^k} + 1 < \frac{n}{2^{d/2}}$.

Remark 1. The algorithm has the special property of being *oblivious* in that the solution maintained, or any other part of memory, is never consulted in the operation of the algorithm until it is output.

The utility of advice. When both n and p are known in advance, we can obtain from the Basic schema an algorithm that requires only logarithmic space in addition to the solution bits. A single bit can record whether a node is contained in the current independent set solution, i.e. in $\overline{S} \cap X$. If $p = 1/d^*$, the reciprocal of the efficient degree, then Lemma 1 yields the following bound that is slightly weaker than $i(H)$.

Theorem 6. *When n and p are known in advance, there is an online streaming algorithm that finds an independent set of expected weight $\Omega(n/d^*)$ in $O(\log n)$ extra space and using $O(r)$ time to process each edge.*

In comparison with the earlier zero-space algorithms, this suggests that knowledge of the input parameters is highly useful for IS. This relates to the recent *annotation model* of [6], although the assumption there is that the advice is dispensed only after the stream is given.

6 Open Questions

What is the right model for graph problems in the streaming context? All of our algorithms for IS use: A read-once-only tape + A tape for the output stream with very limited access + Poly-logarithmic work space. Is poly-logarithmic work space + restricted storage types the way to capture a range of graph problems that do not fit conveniently into existing streaming models? If other graph problems can be also captured by this model, this could grow into a new brand of research.

It would be interesting to compute the constants hidden in our performance gaurantees. Finally, we conjecture that we can make all of our algorithms run without advance knowledge of n.

Acknowledgement. We are grateful to Páll Melsted for helpful discussions.

References

1. Ahn, K.J., Guha, S.: Graph Sparsification in the Semi-streaming Model. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5556, pp. 328–338. Springer, Heidelberg (2009)
2. Alon, N., Arad, U., Azar, Y.: Independent Sets in Hypergraphs with Applications to Routing via Fixed Paths. In: Hochbaum, D.S., Jansen, K., Rolim, J.D.P., Sinclair, A. (eds.) RANDOM 1999 and APPROX 1999. LNCS, vol. 1671, pp. 16–27. Springer, Heidelberg (1999)
3. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. J. Computer and System Sciences 58(1), 1167–1181 (1999)
4. Alon, N., Spencer, J.H.: The probabilistic method. John Wiley and Sons, Chichester (1992)
5. Bar-Yossed, Z., Kumar, R., Sivakumar, D.: Reductions in streaming algorithms, with an application to counting triangles in graphs. In: SODA, pp. 623–632 (2002)
6. Cormode, G., Mitzenmacher, M., Thaler, J.: Streaming graph computations with a helpful advisor. arXiv:1004.2899v1
7. Demetrescu, C., Finocchi, I., Ribichini, A.: Trading off space for passes in graph streaming problems. In: SODA, pp. 714–723 (2006)
8. Emek, Y., Halldórsson, M.M., Mansour, Y., Patt-Shamir, B., Rawitz, D.: Online set packing. To appear in PODC (2010)
9. Epstein, L., Levin, A., Mestre, J., Segev, D.: Improved approximation guarantees for weighted matching in the semi-streaming model. In: STACS (2010)
10. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On Graph Problems in a Semi-Streaming Model. Theoretical Computer Science 348(2), 207–216 (2005)
11. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: Graph distances in the data-stream model. SIAM J. Comput. 38(5), 1709–1727 (2008)
12. Halldórsson, M.M., Iwama, K., Miyazaki, S., Taketomi, S.: Online independent sets. Theoretical Computer Science 289(2), 953–962 (2002)
13. Muthukrishnan, S.: Data Streams: Algorithms and Applications (2003) (manuscript), http://athos.rutgers.edu/~muthu/stream-1-1.ps
14. Shachnai, H., Srinivasan, A.: Finding Large Independent Sets of Hypergraphs in Parallel. SIAM J. Discrete Math. 18(3), 488–500 (2005)

Preprocessing of Min Ones Problems: A Dichotomy

Stefan Kratsch and Magnus Wahlström

Max-Planck-Institut für Informatik, Saarbrücken, Germany
{skratsch,wahl}@mpi-inf.mpg.de

Abstract. Min Ones Constraint Satisfaction Problems, i.e., the task of finding a satisfying assignment with at most k true variables (Min Ones SAT(Γ)), can express a number of interesting and natural problems. We study the preprocessing properties of this class of problems with respect to k, using the notion of kernelization to capture the viability of preprocessing. We give a dichotomy of Min Ones SAT(Γ) problems into admitting or not admitting a kernelization with size guarantee polynomial in k, based on the constraint language Γ. We introduce a property of boolean relations called *mergeability* that can be easily checked for any Γ. When all relations in Γ are mergeable, then we show a polynomial kernelization for Min Ones SAT(Γ). Otherwise, any Γ containing a non-mergeable relation and such that Min Ones SAT(Γ) is NP-complete permits us to prove that Min Ones SAT(Γ) does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly, by a reduction from a particular parameterization of Exact Hitting Set.

1 Introduction

Preprocessing and data reduction are ubiquitous, especially in the context of combinatorially hard problems. This contrasts the well-known fact that there can be no polynomial-time algorithm that provably shrinks every instance of an NP-hard problem, unless P = NP; we cannot expect every instance to become smaller. An immediate answer to this apparent disconnect between theory and practice is to consider preprocessing to be a heuristic, not open to theoretical performance guarantees, but this is not necessary. A rigorous study is possible, but one has to accept the existence of instances that cannot be reduced any further. Such instances can be thought of as already having small size compared to their inherent combinatorial complexity. In the light of this we should measure the success of preprocessing related to the difficulty of the instances.

This new goal can be formulated nicely using notions from parameterized complexity. An instance of a parameterized problem, say (I, k), features an additional parameter, intended to capture the super-polynomial part of the combinatorial complexity. Thus we can define polynomial-time preprocessing in a way that cannot be simply ruled out under the assumption that P $\neq$ NP: a so-called *polynomial kernelization* is a polynomial-time mapping $K : (I, k) \mapsto (I', k')$ such that (I, k) and (I', k') are equivalent and k' as well as the size of I' are bounded

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 653–665, 2010.

by a polynomial in the parameter k. While a kernelization with some performance guarantee can be obtained by various techniques, achieving the strongest size bounds or at least breaking the polynomial barrier is of high interest. Consider for example the improvements for Feedback Vertex Set, from the first polynomial kernel [6], to cubic [3], and now quadratic [21]; the existence of a linear kernel is still an open problem.

Recently a seminal paper by Bodlaender, Downey, Fellows, and Hermelin [4] provided a framework to rule out polynomial kernelizations, based on hypotheses in classical complexity. Using results by Fortnow and Santhanam [12], they showed that so-called *compositional* parameterized problems admit no polynomial kernelizations unless NP $\subseteq$ co-NP/poly; by Yap [22], this would imply that the polynomial hierarchy collapses. The existence of such lower bounds has sparked high activity in the field.

Constraint satisfaction problems. CSPs are a fundamental and general problem setting, encompassing a wide range of natural problems, e.g., satisfiability, graph modification, and covering problems. A CSP instance is a formula given as a conjunction of restrictions, called *constraints*, on the feasible assignments to a set of variables. The complexity of a CSP problem depends on the type of constraints that the problem allows. For instance, Clique can be considered as a maximization problem (Max Ones; see below) allowing only constraints of the type $(\neg x \vee \neg y)$. The types of constraints allowed in a problem are captured by a *constraint language* Γ (see Section 2 for definitions).

There are several results that characterize problem properties depending on the constraint language. In 1978, Schaefer [20] classified the problem SAT(Γ) of deciding whether any satisfying assignment exists for a formula as being either in P or NP-complete; as there are no intermediate cases, this is referred to as a *dichotomy*. Khanna et al. [14] classified CSPs according to their approximability, for the problems Min/Max Ones (i.e., finding a satisfying assignment of optimal weight) and Min/Max SAT (i.e., optimizing the number of satisfied or unsatisfied constraints). On the parameterized complexity side, Marx [18] classified the problem of finding a solution with *exactly* k true variables as being either FPT or W[1]-complete. In this paper we are concerned with the problem family Min Ones SAT(Γ):

> **Input:** A formula $\mathcal{F}$ over a finite constraint language Γ; an integer k.
> **Parameter:** k.
> **Task:** Decide whether there is a satisfying assignment for $\mathcal{F}$ with at most k true variables.

We study the kernelization properties of Min Ones SAT(Γ), parameterized by the maximum number of true variables, and classify these problems into admitting or not admitting a polynomial kernelization. Note that Min Ones SAT(Γ) is in FPT for every finite Γ, by a simple branching algorithm [18]. We also point out that Max SAT(Γ), as a subset of Max SNP (cf. [14]), admits polynomial kernelizations for any constraint language Γ [15].

Related work. In the literature there exists an impressive list of problems that admit polynomial kernels (in fact often linear or quadratic); giving stronger and stronger kernels has become its own field of interest. We name only a few results for problems that also have a notion of arity: a kernelization achieving $O(k^{d-1})$ universe size for Hitting Set with set size at most d [1], a kernelization to $O(k^{d-1})$ vertices for packing k vertex disjoint copies of a d-vertex graph [19], and kernelizations to $O(k^d)$ respectively $O(k^{d+1})$ base set size for any problem from MIN $\mathrm{F}^+\Pi_1$ or MAX NP with at most d variables per clause [15].

Let us also mention a few lower bound results that are based on the framework of Bodlaender et al. [4]. First of all, Bodlaender et al. [5] provided kernel-preserving reductions, which can be used to extend the applicability of the lower bounds; a similar reduction concept was given by Harnik and Naor [13].

Using this, Dom et al. [9] gave polynomial lower bounds for a number of problems, among them Steiner Tree and Connected Vertex Cover. Furthermore they considered problems that have a $k^{f(d)}$ kernel, where k is the solution size and d is a secondary parameter (e.g., maximum set size), and showed that there is no kernel with size polynomial in $k + d$; for, e.g., Hitting Set, Set Cover, and Unique Coverage. More recently, Dell and van Melkebeek [8] provided polynomial local bounds for a list of problems, implying that d-Hitting Set, for $d \geq 2$, cannot have a kernel with *total* size $O(k^{d-\epsilon})$ for any $\epsilon > 0$ unless the polynomial hierarchy collapses. (Note that this is a bound on total size, i.e., the space required to write down the instance, while the previously cited upper bounds are on the number of vertices.) In [16] the present authors show that a certain Min Ones SAT problem does not admit a polynomial kernel and employ this bound to show that there are $\mathcal{H}$-free edge deletion respectively edge editing problems that do not admit a polynomial kernel.

Our work. We give a complete classification of Min Ones SAT(Γ) problems with respect to polynomial kernelizability. We introduce a property of boolean relations called *mergeability* (see Section 2) and, apart from the hardness dichotomy of Theorem 2, we distinguish constraint languages Γ by being *mergeable* or containing at least one relation that is not mergeable. We prove the following result.

Theorem 1. *For any finite constraint language Γ, Min Ones SAT(Γ) admits a polynomial kernelization if it is in* P *or if all relations in Γ are mergeable. Otherwise it does not admit a polynomial kernelization unless* NP $\subseteq$ *co*-NP*/poly.*

When all relations in Γ are mergeable we are able to provide a new polynomial kernelization based on *sunflowers* and *non-zero-closed cores* (Section 3). We say that constraints form a sunflower when they have the same variables in some positions, the *core*, and the variable sets of the other positions, the *petals*, are disjoint (an analogue of sunflowers from extremal set theory). By an adaption of the Erdős-Rado Sunflower Lemma [11] such sunflowers can be easily found in sets containing more than $O(k^d)$ constraints. If we could then replace these sunflowers by smaller, equivalent constraints (e.g., of lower arity), we would be done. Unlike in the d-Hitting Set case (where this is trivial), this is not always possible in general. However, we show that for mergeable constraints forming

a sunflower there is an equivalent replacement that increases the number of *zero-closed* positions; a position of a constraint is *zero-closed* if changing the corresponding variable to zero cannot turn the constraint false. We show how to find sunflowers such that this replacement leads to a simplification of the instance, resulting in a polynomial-time kernelization with $O(k^{d+1})$ variables.

If at least one relation in Γ is not mergeable and Min Ones SAT(Γ) is NP-complete by [14] then we show, using Schaefer-style implementations, that this allows us to implement an n-ary *selection formula* using $O(\log n)$ true local variables (Section 4). A selection formula is simply any kind of formula that is false when no variables are true, and true whenever exactly one variable is true, with arbitrary behavior otherwise; particular examples are disjunctions and parity checks. We show that log-cost selection formulas essentially suffice to allow a kernelization-preserving reduction from Exact Hitting Set (parameterized by the number of sets), and that this problem does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly (in a proof following Dom et al. [9]).

2 CSPs and Mergeability

A *constraint* $R(x_1, \ldots, x_d)$ is the application of a relation R to a tuple of variables; a *constraint language* Γ is a set of relations. A *formula over* Γ is a conjunction of constraints using relations $R \in \Gamma$. We consider only finite constraint languages, and only boolean variables. Finally, having fixed a constraint language Γ, a CSP instance is a formula $\mathcal{F}$ over Γ. The problem Min Ones SAT(Γ) was defined in the previous section. As a technical point, we allow repeated variables in our constraints (e.g., $R(x, x, y)$ is allowed).

The NP-hardness of Min Ones SAT(Γ) was characterized in [14]. For the statement of the theorem, we need to define a few basic relation types. A relation R is *0-valid* if $(0, \ldots, 0) \in R$; *Horn* if it can be expressed as a conjunction of clauses containing at most one positive literal each; and *width-2 affine* if it can be expressed as a conjunction of constraints of the form $x = y$, $x \neq y$, and $x = b$ for $b \in \{0, 1\}$. A constraint language Γ is 0-valid (Horn, width-2 affine) if every $R \in \Gamma$ is. Then, the results of [14] are as follows.

Theorem 2 ([14]). *Let Γ be a finite set of relations over the boolean domain. If Γ is zero-valid, Horn, or width-2 affine,then Min Ones SAT(Γ) is in* P *(even with non-negative weights on the variables); otherwise it is* NP*-complete.*

We now define mergeability and give some basic results about it. First, we need some notation: For two tuples $\alpha = (\alpha_1, \ldots, \alpha_r)$, $\beta = (\beta_1, \ldots, \beta_r)$, let $\alpha \wedge \beta = (\alpha_1 \wedge \beta_1, \ldots, \alpha_r \wedge \beta_r)$, and likewise for $\alpha \vee \beta$, and write $\alpha \leq \beta$ if $\alpha_i \leq \beta_i$ for every $1 \leq i \leq r$ (where 0 and 1 are used for false and true values, respectively). Let Γ be a finite set of relations over the boolean domain. We say that Γ *implements* a relation R if R is the set of satisfying assignments for a formula over Γ, i.e., $R(x_1, \ldots, x_r) \equiv \bigwedge_i R_i(x_{i1}, \ldots, x_{it})$ where each $R_i \in \Gamma$ (we do not automatically allow the equality relation unless $=\in \Gamma$). A constraint is *IHSB-* (Implicative Hitting Set Bounded-) if it can be implemented by assignments,

implications, and negative clauses (i.e., disjunctions of negated variables). Constraint types can also be characterized by closure properties. A constraint R is IHSB- if and only if it is closed under an operation $\alpha \wedge (\beta \vee \gamma)$ for tuples α, β, γ in R, i.e., $\alpha \wedge (\beta \vee \gamma) \in R$ for any choice of $\alpha, \beta, \gamma \in R$. See [7] for more on this.

Definition 1. *Let R be a relation on the boolean domain. Given four (not necessarily distinct) tuples $\alpha, \beta, \gamma, \delta \in R$, we say that the* merge operation *applies if $\alpha \wedge \delta \leq \beta \leq \alpha$ and $\beta \wedge \gamma \leq \delta \leq \gamma$. If so, then* applying *the merge operation produces the tuple $\alpha \wedge (\beta \vee \gamma)$. We say that R is* mergeable *if for any four tuples $\alpha, \beta, \gamma, \delta \in R$ for which the merge operation applies, we have $\alpha \wedge (\beta \vee \gamma) \in R$.*

We give some basic results about mergeability: an alternate presentation of the property, which will be important in Section 3 when sunflowers are introduced, and some basic closure properties. Both propositions are easy to check.

Proposition 1. *Let R be a relation of arity r on the boolean domain. Partition the positions of R into two sets, called the* core *and the* petals*; w.l.o.g. assume that positions 1 through c are the core, and the rest the petals. Let (α_C, α_P), where α_C is a c-ary tuple and α_P an $(r - c)$-ary tuple, denote the tuple whose first c positions are given by α_C, and whose subsequent positions are given by α_P. Consider then the following four tuples.*

$$\begin{aligned} \alpha &= (\alpha_C, \alpha_P) \\ \beta &= (\alpha_C, 0) \\ \gamma &= (\gamma_C, \gamma_P) \\ \delta &= (\gamma_C, 0) \end{aligned}$$

If α through δ are in R, then the merge operation applies, giving us

$$(\alpha_C, \alpha_P \wedge \gamma_P) \in R.$$

Furthermore, for any four tuples to which the merge operation applies, there is a partitioning of the positions into core and petals such that the tuples can be written in the above form.

Proposition 2. *Mergeability is preserved by assignment and identification of variables, i.e., if R is mergeable, then so is any relation produced from R by these operations. Further, any relation implementable by mergeable relations is mergeable.*

Lemma 1 (⋆[1]). *Any zero-valid relation R which is mergeable is also IHSB-, and can therefore be implemented using negative clauses and implications.*

Examples 1. As basic examples, positive and negative clauses, and implications are mergeable. Thus, the same holds for anything implementable by such relations, such as monotone or antimonotone relations (i.e., $\alpha \in R$ implies $\beta \in R$ for all $\beta \geq \alpha$ resp. $\beta \leq \alpha$). A further positive example is the 3-ary ODD relation $(x \oplus y \oplus z = 1)$, but not the 3-ary EVEN or 4-ary ODD relations.

[1] Proof deferred to the full version [17].

3 Kernelization

In this section we show that Min Ones SAT(Γ) admits a polynomial kernelization if all relations in Γ are mergeable. For the purpose of describing our kernelization we first define a sunflower of tuples, similarly to the original sunflower definition for sets. Then we give an adaption of Erdős and Rado's Sunflower Lemma [11] to efficiently find such structures in the given set of constraints. We point out that a similar though more restricted definition for sunflowers of tuples was given by Marx [18]; accordingly the bounds of our sunflower lemma are considerably smaller.

Definition 2. *Let $\mathcal{U}$ be a finite set, let $d \in \mathbb{N}$, and let $\mathcal{H} \subseteq \mathcal{U}^d$. A* sunflower *(*of tuples*)* with cardinality t and core $C \subseteq \{1, \ldots, d\}$ *in $\mathcal{U}$ is a subset consisting of t tuples that have the same element at all positions in C and, in the remaining positions, no element occurs in more than one tuple. The set of remaining positions $P = \{1, \ldots, d\} \setminus C$ is called the* petals.

As an example, $(x_1, \ldots, x_c, y_{11}, \ldots, y_{1p})$, $\ldots$, $(x_1, \ldots, x_c, y_{t1}, \ldots, y_{tp})$ is a sunflower of cardinality t with core $C = \{1, \ldots, c\}$, if all y_{ij} and $y_{i'j'}$ are distinct when $i \neq i'$. Note that, differing from Marx [18] variables in the petal positions may also occur in the core. Observe that every feasible assignment of weight less than t must assign 0 to all variables $y_{i1}, \ldots, y_{ip}$ for some $i \in \{1, \ldots, t\}$. Thus such an assignment must satisfy $R(x_1, \ldots, x_c, 0, \ldots, 0)$ when there are constraints $R(x_1, \ldots, x_c, y_{11}, \ldots, y_{1p})$, $\ldots$, $R(x_1, \ldots, x_c, y_{t1}, \ldots, y_{tp})$.

Lemma 2 ($\star$). *Let $\mathcal{U}$ be a finite set, let $d \in \mathbb{N}$, and let $\mathcal{H} \subseteq \mathcal{U}^d$. If the size of $\mathcal{H}$ is greater than $k^d(d!)^2$, then a sunflower of cardinality $k+1$ in $\mathcal{H}$ can be efficiently found.*

We require some definitions revolving around sunflowers and zero-closed positions to address the issue of simplifying constraints that form a sunflower.

Definition 3. *Let R be an r-ary relation. The relation R is* zero-closed on position i*, if for every tuple $(t_1, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_r) \in R$ we have that R contains also $(t_1, \ldots, t_{i-1}, 0, t_{i+1}, \ldots, t_r)$. A relation is* non-zero-closed *if it has no zero-closed positions. We define functions Δ, Π, and ∇:*

- *$\Delta_P(R)$ is defined to be the* zero-closure of R on positions $P \subseteq \{1, \ldots, r\}$*, i.e., the smallest superset of R that is zero-closed on all positions $i \in P$.*
- *$\Pi(R)$ denotes the* non-zero-closed core*, i.e., the projection of R onto all positions that are not zero-closed or, equivalently, the relation on the non-zero-closed positions obtained by forcing $x_i = 0$ for all zero-closed positions.*
- *$\nabla_C(R)$ denotes the* sunflower restriction of R with core C*: w.l.o.g. the relation given by $[\nabla_C(R)](x_1, \ldots, x_r) = R(x_1, \ldots, x_r) \wedge R(x_1, \ldots, x_c, 0, \ldots, 0)$ for $C = \{1, \ldots, c\}$.*

Variables occurring only in zero-closed positions are easy to handle since setting them to zero does not restrict the assignment to any other variable. To measure the number of non-zero-closed positions in our formula, i.e., occurrences of variables that prevent us from such a replacement, we introduce $\mathcal{Z}(\mathcal{F}, R)$.

Definition 4. *Let $\mathcal{F}$ be a formula and let R be a relation. We define $\mathcal{Z}(\mathcal{F}, R)$ as the set of all tuples $(x_1, \ldots, x_t)$ where $[\Pi(R)](x_1, \ldots, x_t)$ is the non-zero-closed core of an R-constraint in $\mathcal{F}$.*

We prove that mergeable relations admit an implementation of their sunflower restrictions using the respective zero-closures on the petal positions. By choosing carefully the sunflowers that we replace such that the petals contain non-zero-closed positions this will lower $|\mathcal{Z}(\mathcal{F}, R)|$.

Lemma 3 ($\star$). *Let R be a mergeable relation and let $C \cup P$ be a partition of its positions. Then $\Delta_P(\nabla_C(R))$ is mergeable and there is an implementation of $\nabla_C(R)$ using $\Delta_P(\nabla_C(R))$ and implications.*

In the following theorem we establish a preliminary kernel in the form of a formula $\mathcal{F}'$ over some larger constraint language Γ', such that $\mathcal{Z}(\mathcal{F}', R)$ is small for every non-zero-valid relation $R \in \Gamma'$. The language Γ' admits us to apply Lemma 3, as $\Delta_P(\nabla_C(R))$ is not necessarily contained in Γ.

Theorem 3 ($\star$). *Let Γ be a mergeable constraint language with maximum arity d. Let $\mathcal{F}$ be a formula over Γ and let k be an integer. In polynomial time one can compute a formula $\mathcal{F}'$ over a mergeable constraint language $\Gamma' \supseteq \Gamma$ with maximum arity d, such that every assignment of weight at most k satisfies $\mathcal{F}$ if and only if it satisfies $\mathcal{F}'$ and, furthermore, $|\mathcal{Z}(\mathcal{F}', R)| \in O(k^d)$ for every non-zero-valid relation that occurs in $\mathcal{F}'$.*

Proof (sketch). We construct $\mathcal{F}'$, starting from $\mathcal{F}' = \mathcal{F}$. While $|\mathcal{Z}(\mathcal{F}', R)| > k^d(d!)^2$ for any non-zero-valid relation R in $\mathcal{F}'$, search for a sunflower of cardinality $k+1$ in $\mathcal{Z}(\mathcal{F}', R)$, according to Lemma 2. Let C denote the core of the sunflower. Replace each R-constraint whose non-zero-closed core matches a tuple of the sunflower by its sunflower restriction with core C using an implementation according to Lemma 3. Repeating this step until $|\mathcal{Z}(\mathcal{F}', R)| \leq k^d(d!)^2$ for all non-zero-valid relations R in $\mathcal{F}'$ completes the construction.

To show correctness we consider a single replacement. We denote the tuples of the sunflower by $(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip})$, with $i \in \{1, \ldots, k+1\}$, i.e., w.l.o.g. with core $C = \{1, \ldots, c\}$ and petals $P = \{c+1, \ldots, c+p\}$. Let ϕ be any satisfying assignment of weight at most k and consider any tuple $(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip})$ of the sunflower. Let $R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, z_1, \ldots, z_t)$ be a constraint whose non-zero-closed core matches the tuple, w.l.o.g. the last positions of R are zero-closed, let Z be those positions. Since the z_i are in zero-closed positions, ϕ must satisfy $R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, 0, \ldots, 0)$. Observe that, by maximum weight k, the assignment ϕ assigns 0 to all variables $y_{i1}, \ldots, y_{ip}$ for an $i \in \{1, \ldots, k+1\}$. Thus ϕ satisfies $R(x_1, \ldots, x_c, 0, \ldots, 0)$.

Hence for any constraint $R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, z_1, \ldots, z_t)$, the assignment satisfies $\nabla_C(R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, z_1, \ldots, z_t))$ too. This permits us to replace each R-constraint, whose non-zero-closed core matches a tuple of the sunflower, by an implementation of its sunflower restriction with core C, according to Lemma 3. This uses $\Delta_{P \cup Z}(\nabla_C(R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, z_1, \ldots, z_t)))$ and implications. □

Now we are able to establish polynomial kernelizations for Min Ones SAT(Γ) when Γ is mergeable. For a given instance $(\mathcal{F}, k)$, we first generate an equivalent formula $\mathcal{F}'$ according to Theorem 3. However, $\mathcal{F}'$ will not replace $\mathcal{F}$, rather, it allows us to remove variables from $\mathcal{F}$ based on conclusions drawn from $\mathcal{F}'$.

Theorem 4 ($\star$). *For any mergeable constraint language Γ, Min Ones SAT(Γ) admits a polynomial kernelization.*

Proof (sketch). Let $(\mathcal{F}, k)$ be an instance of Min Ones SAT(Γ) and let d be the maximum arity of relations in Γ. According to Theorem 3, we generate a formula $\mathcal{F}'$, such that assignments of weight at most k are satisfying for $\mathcal{F}$ if and only if they are satisfying for $\mathcal{F}'$. Moreover, for each non-zero-valid relation R, we have that $|\mathcal{Z}(\mathcal{F}', R)| \in O(k^d)$. We allow the constant 0 to be used for replacing variables; a simple construction without using $(x = 0)$ is given in the full proof.

First, according to Lemma 1, we replace each zero-valid constraint of $\mathcal{F}'$ by an implementation through negative clauses and implications. Variables that occur only in zero-closed positions in $\mathcal{F}'$ are replaced by 0, without affecting the possible assignments for the other variables. By equivalence of $\mathcal{F}$ and $\mathcal{F}'$ with respect to assignments of weight at most k, the same is true for $\mathcal{F}$.

Let X be the set of variables that occur in a non-zero-closed position of some non-zero-valid constraint of $\mathcal{F}'$. If a variable $x \in X$ implies at least k other variables, i.e., they have to take value 1 if $x = 1$ by implication constraints in $\mathcal{F}'$, then there is no satisfying assignment of weight at most k for $\mathcal{F}'$ that assigns 1 to x. By equivalence of $\mathcal{F}$ and $\mathcal{F}'$ with respect to such assignments, we replace all such variables by 0. Finally we replace all variables $y \in V(\mathcal{F}') \setminus X$, that are not implied by a variable from X in $\mathcal{F}'$, by the constant 0 in $\mathcal{F}$ and $\mathcal{F}'$. Note that such variables occur only in zero-closed positions and in implications.

Now we prove a bound of $O(k^{d+1})$ on the number of variables in $\mathcal{F}$. First, we observe that all remaining variables of $\mathcal{F}$ must occur in a non-zero-closed position of some constraint of $\mathcal{F}'$. We begin by bounding the number of variables that occur in a non-zero-closed position of some non-zero-valid R-constraint, i.e., the remaining variables of the set X. Observe that such a variable must occur in the corresponding tuple of $\mathcal{Z}(\mathcal{F}', R)$. Since there is only a constant number of relations of arity at most d and since $\mathcal{Z}(\mathcal{F}', R) \in O(k^d)$, this limits the number of such variables by $O(k^d)$. For all other variables, their non-zero-closed occurrences must be in implications, since negative clauses are zero-closed on all positions. Thus, these variables must be implied by a variable of X. Since each variable implies at most $k-1$ other variables, we get an overall bound of $O(k^{d+1})$. Finally,

the total size of $\mathcal{F}$ is polynomial for a fix d, since the number of variables is polynomial and the arity of the constraints is bounded. □

4 Kernel Lower Bounds

We will now complete the dichotomy by showing that if Min Ones SAT(Γ) is NP-complete and some $R \in \Gamma$ is not mergeable, then the problem admits no polynomial kernelization unless NP $\subseteq$ co-NP/poly. The central concept of our lower bound construction is the following definition.

Definition 5. *A* selection formula *of arity n is a formula on variable sets X and Y, with $|Y| = n$ and $|X| = n^{O(1)}$, such that there is no solution where $Y = 0$, but for any $y \in Y$ there is a solution where $y = 1$ and $y' = 0$ for any $y' \neq y$, $y' \in Y$; refer to such a solution as* selecting y. *Let the* selection cost *for $y \in Y$ be the minimum number of true variables in X among assignments selecting y. A* log-cost selection formula *is a selection formula where there is a number $w_n = O(\log n)$ such that all selection costs are exactly w_n, and where every solution has at least w_n true variables among X.*

We will show that any Γ as described can be used to construct log-cost selection formulas, and then derive a lower bound from this. The next lemma describes our constructions.

Lemma 4. *The following types of relations can implement log-cost selection formulas of any arity.*
1. A 3-ary relation R_3 with $\{(0,0,0),(1,1,0),(1,0,1)\} \subseteq R_3$ and $(1,0,0) \notin R_3$, together with relations $(x = 1)$ and $(x = 0)$.
2. A 5-ary relation R_5 with $\{(1,0,1,1,0),(1,0,0,0,0),(0,1,1,0,1),(0,1,0,0,0)\} \subseteq R_5$ and $(1,0,1,0,0),(0,1,1,0,0) \notin R_5$, together with relations $(x \neq y)$, $(x = 1)$, and $(x = 0)$.

Proof. Let $Y = \{y_1, \ldots, y_n\}$ be the variables over which a log-cost selection formula is requested. We will create "branching trees" over variables $x_{i,j}$ for $0 \leq i \leq \log_2 n$, $1 \leq j \leq 2^i$, as variants of the composition trees used in [16]. Assume that $n = 2^h$ for some integer h; otherwise pad Y with variables forced to be false, as assumed to be possible in both constructions.

The first construction is immediate. Create the variables $x_{i,j}$ and add a constraint $(x_{0,1} = 1)$. Further, for all i, j with $0 \leq i < h$ and $1 \leq j \leq 2^i$, add a constraint $R_3(x_{i,j}, x_{i+1,2j-1}, x_{i+1,2j})$. Finally, replace variables $x_{h,j}$ by y_j. It can be easily checked that the requirements on R_3 imply that the result is a correct log-cost selection formula with $w_n = h = \log_2 n$.

The second construction uses the same principle, but the construction is somewhat more involved. Create variables $x_{i,j}$ and a constraint $(x_{0,1} = 1)$ as before. In addition, introduce for every $0 \leq i \leq h - 1$ two variables l_i, r_i and a constraint $(l_i \neq r_i)$. Now the intention is that (l_i, r_i) decides whether the path of true variables from the root to a leaf should take a left or a right turn after

level i. Concretely, add for every i, j with $0 \leq i \leq h-1$ and $1 \leq j \leq 2^i$ a constraint $R_5(l_i, r_i, x_{i,j}, x_{i+1,2j-1}, x_{i+1,2j})$. It can again be verified that this creates a log-cost selection formula with $w_n = 2h = 2\log_2 n$. □

We now reach the technical part, where we show that any relation which is not mergeable can be used to construct a relation as in Lemma 4. The constructions are based on the concept of a *witness* that some relation R lacks a certain closure property. For instance, if R is not mergeable, then there are four tuples $\alpha, \beta, \gamma, \delta \in R$ to which the merge operation applies, but such that $\alpha \wedge (\beta \vee \gamma) \notin R$; these four tuples form a witness that R is not mergeable. Using the knowledge that such witnesses exist, we can use the approach of Schaefer [20], identifying variables according to their occurrence in the tuples of the witness, to build relations with the properties we need.

Lemma 5 (⋆). *Let Γ be a set of relations such that Min Ones SAT(Γ) is* NP*-complete and some $R \in \Gamma$ is not mergeable. Under a constraint that at most k variables are true, Γ can be used to force $(x = 0)$ and $(x = 1)$. Furthermore, there is an implementation of $(x = y)$ using R, $(x = 0)$, and $(x = 1)$.*

Lemma 6 (⋆). *Let Min Ones SAT(Γ) be NP-complete, and not mergeable. Then Min Ones SAT(Γ) can express a log-cost selection formula of any arity.*

We now show our result, using the tools of [5]. We have the following definition. Let $\mathcal{Q}$ and $\mathcal{Q}'$ be parameterized problems. A *polynomial time and parameter transformation* from $\mathcal{Q}$ to $\mathcal{Q}'$ is a polynomial-time mapping $H : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N} : (x, k) \mapsto (x', k')$ such that

$$\forall (x, k) \in \Sigma^* \times \mathbb{N} : ((x, k) \in \mathcal{Q} \Leftrightarrow (x', k') \in \mathcal{Q}') \text{ and } k' \leq p(k),$$

for some polynomial p.

We will provide a polynomial time and parameter transformation to Min Ones SAT(Γ) from Exact Hitting Set(m), defined as follows.

> **Input:** A hypergraph $\mathcal{H}$ consisting of m subsets of a universe U of size n.
> **Parameter:** m.
> **Task:** Decide whether there is a set $S \subset U$ such that $|E \cap S| = 1$ for every $E \in \mathcal{H}$.

It was shown in [5] that polynomial time and parameter transformations preserve polynomial kernelizability, thus our lower bound will follow. We first show that Exact Hitting Set(m) admits no polynomial kernelization; the proof follows the outline of Dom et al. [9].

Lemma 7 (⋆). *Exact Hitting Set(m) admits no polynomial kernelization unless* NP $\subseteq$ *co-*NP*/poly.*

We can now show the main result of this section.

Theorem 5. *Let Γ be a constraint language which is not mergeable. Then Min Ones SAT(Γ) is either in* P*, or does not admit a polynomial kernelization unless* NP $\subseteq$ *co-*NP*/poly.*

Proof. By Theorem 2, Min Ones SAT(Γ) is either polynomial-time solvable or NP-complete; assume that it is NP-complete. By Lemma 5 we have both constants and the constraint $(x = y)$, and by Lemma 6 we can implement log-cost selection formulas. It remains only to describe the polynomial time and parameter transformation from Exact Hitting Set(m) to Min Ones SAT(Γ).

Let $\mathcal{H}$ be a hypergraph. If $\mathcal{H}$ contains more than 2^m vertices, then it can be solved in time polynomial in the input length [2]; otherwise, we create a formula $\mathcal{F}$ and fix a weight k so that $(\mathcal{F}, k)$ is positive if and only if $\mathcal{H}$ has an exact hitting set. Create one variable $y_{i,j}$ in $\mathcal{F}$ for every occurrence of a vertex v_i in an edge E_j in $\mathcal{H}$. For each edge $E \in \mathcal{H}$, create a selection formula over the variables representing the occurrences in E. Finally, for all pairs of occurrences of each vertex v_i, add constraints $(y_{i,j} = y_{i,j'})$, and fix $k = m + \sum_{E \in \mathcal{H}} w_{|E|}$, where w_i is the weight of an i-selection formula. We have an upper bound on the value of k of $O(m \log n) = O(m^2)$.

Now solutions with weight exactly k correspond to exact hitting sets of $\mathcal{H}$. Note that k is the minimum possible weight of the selection formulas, which is taken if exactly one occurrence in each edge is picked. By the definition of log-cost selection formulas, any solution where more than one occurrence has been picked (if such a solution is possible at all) will have a total weight which is larger than this, if the weight of the y-variables is counted as well, and thus such a solution to $\mathcal{F}$ of weight at most k is not possible.

As Exact Hitting Set(m) is NP-complete, it follows from [5] that a polynomial kernelization for Min Ones SAT(Γ) would imply the same for Exact Hitting Set(m), giving our result. □

Finally, let us remark that our lower bound still applies under the restriction that constraints contain no repeated variables. Lemma 5 can be adjusted to provide $(x = 1)$ and $(x = y)$ under this restriction, and we can then use standard techniques (see [16] and Theorem 4) to complete the bound. Such a restriction can be useful in showing hardness of other problems, e.g., as in [16].

5 Conclusions

We presented a dichotomy for Min Ones SAT(Γ) for finite sets of relations Γ, characterized by a new property called mergeability. We showed that Min Ones SAT(Γ) admits a polynomial kernelization if the problem is in P or if every relation in Γ is mergeable, while in every other case no polynomial kernelization is possible, unless NP $\subseteq$ co-NP/poly and the polynomial hierarchy collapses.

An immediate question is the correct size bound for the kernelizable cases. In this paper, the total size is bounded only as a side-effect of having $O(k^{d+1})$ variables, while in [8], it was shown that for a number of problems, including d-Hitting Set, the "correct" bound on the total size of a kernel is $O(k^d)$. Closing this gap, or even characterizing the problems which admit e.g. quadratic total size kernels, would be interesting. Similarly, one can ask whether an explicit super-polynomial (e.g. $2^{(\log m)^{O(1)}}$) or exponential lower bound on the kernelizability

of our source problem Exact Hitting Set (m) is possible. For this, and related questions, a study of the structure of problems in FPT under the closure of kernelization-preserving reductions may be useful.

Another question is how the results extend to problems on larger domains, e.g., when variables can take t different values, but at most k may be non-zero.

Acknowledgements. The authors are thankful to Gustav Nordh and Dániel Marx for helpful and interesting discussions.

References

1. Abu-Khzam, F.N.: Kernelization algorithms for d-hitting set problems. In: Dehne, F., Sack, J.-R., Zeh, N. (eds.) WADS 2007. LNCS, vol. 4619, pp. 434–445. Springer, Heidelberg (2007)
2. Björklund, A., Husfeldt, T.: Exact algorithms for exact satisfiability and number of perfect matchings. Algorithmica 52(2), 226–249 (2008)
3. Bodlaender, H.L.: A cubic kernel for feedback vertex set. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 320–331. Springer, Heidelberg (2007)
4. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels (extended abstract). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 563–574. Springer, Heidelberg (2008)
5. Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 635–646. Springer, Heidelberg (2009)
6. Burrage, K., Estivill-Castro, V., Fellows, M.R., Langston, M.A., Mac, S., Rosamond, F.A.: The undirected feedback vertex set problem has a poly(k) kernel. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 192–202. Springer, Heidelberg (2006)
7. Creignou, N., Vollmer, H.: Boolean constraint satisfaction problems: When does Post's lattice help? In: Creignou, N., Kolaitis, P.G., Vollmer, H. (eds.) Complexity of Constraints. LNCS, vol. 5250, pp. 3–37. Springer, Heidelberg (2008)
8. Dell, H., van Melkebeek, D.: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In: STOC (to appear, 2010)
9. Dom, M., Lokshtanov, D., Saurabh, S.: Incompressibility through colors and ids. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 378–389. Springer, Heidelberg (2009)
10. Downey, R.G., Fellows, M.R.: Parameterized Complexity (Monographs in Computer Science). Springer, Heidelberg (November 1998)
11. Erdős, P., Rado, R.: Intersection theorems for systems of sets. J. London Math. Soc. 35, 85–90 (1960)
12. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. In: STOC, pp. 133–142. ACM, New York (2008)
13. Harnik, D., Naor, M.: On the compressibility of NP instances and cryptographic applications. SIAM J. Comput. 39(5), 1667–1713 (2010)
14. Khanna, S., Sudan, M., Trevisan, L., Williamson, D.P.: The approximability of constraint satisfaction problems. SIAM J. Comput. 30(6), 1863–1920 (2000)

15. Kratsch, S.: Polynomial kernelizations for MIN $F^+\Pi_1$ and MAX NP. In: STACS 2009. Dagstuhl Seminar Proceedings, vol. 09001, pp. 601–612. Schloss Dagstuhl, Germany (2009)
16. Kratsch, S., Wahlström, M.: Two edge modification problems without polynomial kernels. In: Chen, J., Fomin, F.V. (eds.) IWPEC. LNCS, vol. 5917, pp. 264–275. Springer, Heidelberg (2009)
17. Kratsch, S., Wahlström, M.: Preprocessing of min ones problems: a dichotomy. CoRR, abs/0910.4518 (2009)
18. Marx, D.: Parameterized complexity of constraint satisfaction problems. Computational Complexity 14(2), 153–183 (2005)
19. Moser, H.: A problem kernelization for graph packing. In: Nielsen, M., Kucera, A., Miltersen, P.B., Palamidessi, C., Tuma, P., Valencia, F.D. (eds.) SOFSEM 2009. LNCS, vol. 5404, pp. 401–412. Springer, Heidelberg (2009)
20. Schaefer, T.J.: The complexity of satisfiability problems. In: STOC, pp. 216–226. ACM, New York (1978)
21. Thomassé, S.: A quadratic kernel for feedback vertex set. In: SODA, pp. 115–119. SIAM, Philadelphia (2009)
22. Yap, C.-K.: Some consequences of non-uniform conditions on uniform classes. Theor. Comput. Sci. 26, 287–300 (1983)

Holographic Reduction: A Domain Changed Application and Its Partial Converse Theorems

Mingji Xia*

State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences,
P.O. Box 8718, Beijing 100190, China

Abstract. Holographic reductions between some Holant problems and some #CSP(H_d) problems are built, where H_d is some complex value binary function. By the complexity of these Holant problems, for each integer $d \geq 2$, #CSP(H_d) is in P when each variables appears at most d times, while it is #P-hard when each variables appears at most $d+1$ times. #CSP(H_d) counts weighted summation of graph homomorphisms from input graph G to graph H_d, and the maximum occurrence of variables is the maximum degree of G.

We conjecture the converse of holographic reduction holds for most of #Bi-restriction Constraint Satisfaction Problems, which can be regarded as a generalization of a known result about counting graph homomorphisms. It is proved that the converse of holographic reduction holds for some classes of problems.

Keywords: holographic reduction, graph homomorphism, holant problem, #CSP, #BCSP.

1 Introduction

Class #P is proposed by Valiant and permanent is #P-hard [19]. An important kind of complexity results about counting problems in #P is dichotomy theorem, which claims that all problems is a class is either polynomial computable or #P-hard. There are dichotomy theorems for #CSP [12] [2] [14] [3] [10], counting graph homomorphisms [16] [1] [8], and Holant problems [10] [11]. Most studied #CSP problems and Holant problems have domain size 2. Except [15], most studied counting graph homomorphisms problems have arbitrary domain size, and they are #CSP problem defined by one binary function, if we permits self-loop and multi-edge in input graphs. Holant problem is a restricted version of #CSP, such that each variable occurs twice, but this restriction makes it a more general problem than #CSP.

In this paper, we construct a series of complex value binary symmetric functions H_d, $d = 2, 3, \ldots$. For each integer $d \geq 2$, if permitting a variable occurring

* Supported by the Hundred-Talent program of the Chinese Academy of Sciences under Angsheng Li, and the Grand Challenge Program "Network Algorithms and Digital Information" of the Institute of Software, Chinese Academy of Sciences.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 666–677, 2010.

$d+1$ times instead of d times, a polynomial computable #CSP(H_d) problem becomes #P-hard. In the language of counting graph homomorphisms, for each integer $d > 2$, there exists a complex weighted undirected graph H_d, such that counting the summation of the weights of all homomorphisms from input graph G to H_d is #P-hard, when the maximum degree of G is restricted to $d+1$, but it has polynomial time algorithm, when the maximum degree is restricted to d. By the notation #$\mathbf{F}|\mathbf{H}$ of #Bi-restriction Constraint Satisfaction Problem, for each integer $d \geq 2$, $\#\{H_d\}|\{=_1, =_2, \ldots, =_d\}$ is polynomial time computable, while $\#\{H_d\}|\{=_1, =_2, \ldots, =_{d+1}\}$ is #P-hard, where $=_k$ denote the equivalence relation of arity k.

It is well known that #SAT and #2SAT are #P-hard. There are many other #P-hard results of maximum degree bounded counting problems in [18]. There are two general results about maximum degree and complexity for a class of problems. In [16], it is proved that if #CSP(H) is hard, then there exist some constant C (maybe depends on H), such that it is still hard when the maximum degree of input graph G is restricted to C, where H is a 0-1 weighted undirected graph. That is, H is a binary function from $[n]^2$ to $\{0, 1\}$ or rational numbers, where n is the number of vertices in H. In our result, the range of H_d is complex numbers, and we do not know whether it can be strengthened to $\{0, 1\}$. In [10], it is proved for complexity weighted Boolean #CSP, if #CSP($\mathbf{F}$) is hard, then it is also hard, when each variables appears at most 3 times ($\#\mathbf{F}|\{=_1, =_2, =_3\}$). In Boolean #CSP, each variable takes value from Boolean domain $\{0, 1\}$, so our result does not hold for Boolean domain. In our result, the domain of H_d is very large, depending on the construction and d. Our result shows, in a large class of counting problems, the relation of maximum degree and complexity is quite complicated.

This result is proved mainly by holographic reduction from some Boolean domain Holant problems. Holographic reduction is proposed by Valiant in his senior paper holographic algorithms [21]. There have been lots of studies of holographic reduction [22] [23] [4] [5] [6] [7] [24], including designing algorithms on planar graphs, and characterization of matchgates under holographic reduction, and proving #P-hardness, etc. Here the holographic reduction is between two problems of different domains. There are not many this kind of applications. The first example is the holographic algorithm for PL-FO-2-COLOR problem in [21]. Its power is not very clear, although there is characterization for matchgates case [5].

We also study the converse of holographic reduction. It is an algebra problem, asking whether the sufficient condition in holographic reduction is also a necessary condition. We conjecture it holds, since it is a generalization of a known result about graph homomorphisms [17,13], and we prove that it holds for some classes of Holant problems.

In section 2, we introduce definitions and holographic reduction. In section 3, we prove the result about maximum degree and complexity. In section 4, we prove the converse of holographic reduction holds for some classes of Holant problems.

2 Preliminary

Let $[n]$ denote the set $\{0, 1, \ldots, n-1\}$. $[f_0, f_1, \ldots, f_k]$ denotes a symmetric function F over $[2]^k$, such that f_i is the value of F on the inputs of Hamming weight i. The value table of a function F over $[n]^k$ can be written as a column vector $F = (F_{x_1x_2\cdots x_k})$ or a row vector $F' = (F_{x_1x_2\cdots x_k})'$ of length n^k, where $F_{x_1x_2\cdots x_k} = F(x_1, x_2, \ldots, x_k)$. We also look a row or column vector of length n^k as a function in the same way. A binary function $F(x, y)$ can also be written as a matrix $(T_{x,y})$, where $T_{x,y} = F(x, y)$, and this matrix is denoted by $\widehat{F}$.

Let $=_k$ denote the equivalence relation in k variables. For example, $=_1$ is $[1, 1]$,and $=_2$ is $[1, 0, 1]$, when variables are in domain $[2]$. Let $\mathbf{R}^{\mathbf{d}}_{=}$ denote the set $\{=_1, =_2, \ldots, =_d\}$, and $\mathbf{R}_{=}$ denote the set of all equivalence relations.

Define a general counting problem $\#\mathbf{F}|\mathbf{H}$, named #Bi-restriction Constraint Satisfaction Problem. $\mathbf{F}$ and $\mathbf{H}$ are two sets of functions in variables of domain $[n]$.

An instance (G, ϕ_l, ϕ_r) of this problem is a bipartite graph $G(U, V, E)$, and two mappings, $\phi_l : v \in U \to F_v \in \mathbf{F}$ and $\phi_r : v \in V \to F_v \in \mathbf{H}$ (using F_v to denote the value of ϕ_l or ϕ_r on v), satisfying the arity of F_v is d_v, the degree of v. The bipartite graph G is given as two one to one mappings, $\phi_1 : (v, i) \to e$ and $\phi_2 : (u, i) \to e$, where $v \in V$ and $u \in U$ respectively, $i \in [d_v]$ (or $[d_u]$), and $e \in E$ is one of the edges incident to v (or u). Let $e_{v,i} = \phi_s(v, i)$, $v \in U \cup V$, $s = 1, 2$.

In such an instance, edges are looked as variables with domain $[n]$. Vertex v is looked as function F_v(specified by ϕ_l or ϕ_r) in its edges, and $e_{v,i}$ specifies which edge of v is the ith input of F_v.

The value on this instance is defined as a summation over all $[n]$ valued assignments σ of edges,

$$\#\mathbf{F}|\mathbf{H}(G, \phi_l, \phi_r) = \sum_{\sigma: E \to [n]} \prod_{v \in U \cup V} F_v(\sigma(e_{v,1}), \sigma(e_{v,2}), \ldots, \sigma(e_{v,d_v})).$$

Suppose p is a nonzero constant. If $F \in \mathbf{F}$ is replace by pF, then the value of $\#\mathbf{F}|\mathbf{H}$ is simply multiplied by a power of p. Since p does not affect the computational complexity of $\#\mathbf{F}|\mathbf{H}$, we usually ignore it. $\#\mathbf{F}|\{=_2\}$ is also called Holant($\mathbf{F}$) problem in [10]. $\#\mathbf{F}|\mathbf{R}_{=}$ is #CSP($\mathbf{F}$), equivalent to Holant($\mathbf{F} \cup \mathbf{R}_{=}$).

We say two problems $\#\mathbf{F}|\mathbf{H}$ and $\#\mathbf{P}|\mathbf{Q}$ are *result equivalent*, if there are two bijections $\sigma_l : \mathbf{F} \to \mathbf{P}$ and $\sigma_r : \mathbf{H} \to \mathbf{Q}$, such that for any instance (G, ϕ_l, ϕ_r),

$$\#\mathbf{F}|\mathbf{H}(G, \phi_l, \phi_r) = \#\mathbf{P}|\mathbf{Q}(G, \sigma_l \circ \phi_l, \sigma_r \circ \phi_r).$$

Use $\otimes$ to denote tensor product. Suppose A and B are two matrices, and $A = (a_{ij})$ has size $k \times m$.

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \ldots & a_{1m}B \\ a_{21}B & a_{22}B & \ldots & a_{2m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1}B & a_{k2}B & \ldots & a_{km}B \end{pmatrix}.$$

$A^{\otimes r}$ denotes the tensor product of r matrices, that is, $A^{\otimes 1} = A$ and $A^{\otimes(r+1)} = A^{\otimes r} \otimes A$. If $A = (A_1, \ldots, A_m)$, that is, the ith column of A is A_i, then the $i_1 i_2 \cdots i_r$ column of $A^{\otimes r}$ is $A_{i_1} \otimes \cdots \otimes A_{i_r}$ by definition.

Suppose F is a binary function, then $A^{\otimes 2} F$ and $A\widehat{F}A'$ are the column vector form and matrix form of the same function.

We say two problems $\#\mathbf{F}|\mathbf{H}$ and $\#\mathbf{P}|\mathbf{Q}$ are *algebra equivalent*, if there exist a nonsingular matrix M, and two bijections $\sigma_l : \mathbf{F} \to \mathbf{P}$ and $\sigma_r : \mathbf{H} \to \mathbf{Q}$, such that for any function $F \in \mathbf{F}$ and $H \in \mathbf{H}$,

$$\sigma_l(F)' = F' \; M^{\otimes R_F}, \quad \sigma_r(H) = (M^{-1})^{\otimes R_H} \; H,$$

where R_F (resp. R_H) denotes the arity of F (resp. H).

Both result equivalent and algebra equivalent are equivalence relations.

Theorem 1 ([21]). *If $\#\mathbf{F}|\mathbf{H}$ and $\#\mathbf{P}|\mathbf{Q}$ are algebra equivalent, then they are result equivalent under the same bijections σ_l and σ_r.*

By this theorem, if $\#\mathbf{F}|\mathbf{H}$ and $\#\mathbf{P}|\mathbf{Q}$ are algebra equivalent, we can reduce one to the other. This kind of reduction is called holographic reduction. The matrix M in the algebra equivalent is called the base of holographic reduction. There is another form of this theorem, which is convenient for domain size changed applications.

Theorem 2 ([21]). *Suppose F is a function over $[m]^s$, and H is a function over $[n]^t$, and M is a $m \times n$ matrix. Problems $\#\{F'\}|\{M^{\otimes t} H\}$ and $\#\{F' M^{\otimes s}\}|\{H\}$ are result equivalent.*

This theorem also holds for general function sets like Theorem 1.

3 An Application

Let $\{a_i\}$ is Fibonacci sequence, that is, $a_0 = 0$, $a_1 = 1$, $a_{i+2} = a_{i+1} + a_i$. d is an integer no less than 2. Let $F_i = [a_0, a_1, \ldots, a_i]$, $i = 1, 2, \ldots, d$, $F_{d+1} = [a_0, a_1, \ldots, a_d, -2a_d]$.

We need some complexity results for Holant problems. One is that $\#\{=_2\}|\{F_1, \ldots, F_d\}$ is polynomial time computable [9,10]. The other is the following hardness lemma, which is not a straightforward corollary of the dichotomy theorems for Holant^C and Holant* in [10].

Lemma 1. *For any integer $d \geq 2$, $\#\{=_2\}|\{F_1, \ldots, F_{d+1}\}$ is #P-hard.*

Proof. We will reduce the problems in the following list to the problem before it.

$$\begin{aligned} &\#\{F_1, \ldots, F_{d+1}\}|\{=_2\} \\ &\#\{F_1, P, F_3\}|\{=_2\} \\ &\#\{F_1, F_3\}|\{P\} \end{aligned}$$

$$\#\{=_1,=_3\}|\{Q\}$$
$$\#\{=_1,=_2,=_3\}|\{Q\}$$
$$\#\{=_1,=_2,=_3\}|\{Q,=_2\}$$
$$\#\mathbf{R}_=|\{Q\}$$

In the first reduction, the binary function $P = [a_{d-1}, a_d, -2a_d]$ is constructed directly by connecting $d-1$ functions $F_1 = [0,1]$ to F_{d+1}.

The second reduction is simply because $\#\mathbf{F}|\mathbf{F}$ is a restricted version of $\#\mathbf{F}|\{=_2\}$.

The third reduction is holographic reduction by the base $M = \begin{pmatrix} 1 & -1 \\ \frac{1+\sqrt{5}}{2} & \frac{-1+\sqrt{5}}{2} \end{pmatrix}$. $Q = \begin{pmatrix} 2a_d - a_{d-1} & 5a_d + a_{d-1} \\ 5a_d + a_{d-1} & 2a_d - a_{d-1} \end{pmatrix}$, ignoring a constant factor.

In the fourth reduction, $=_2$ is constructed by connecting unary function $\widehat{Q} =_1$ to $=_3$, where $\widehat{Q} =_1$ is constructed by connecting $=_1$ to Q.

In the fifth reduction, we can apply the interpolation reduction method in [16] to realize right side $=_2$, by using Q^s, $s = 1, 2, \ldots, \text{poly(n)}$, to interpolate on the eigenvalues.

In the last reduction, $r-2$ functions $=_3$ are connected by right side $=_2$ to realize left side $=_r$.

All entries in $\widehat{Q}$ is nonzero, and $\widehat{Q}$ is nonsingular, by result in [1], $\#\mathbf{R}_=|\{Q\}$ is #P-hard. □

Theorem 3. *For any integer $d \geq 2$, there exist a complex valued symmetric binary function H_d, such that $\#\{H_d\}|\mathbf{R}^{\mathbf{d}}_{=}$ has polynomial time algorithm, but $\#\{H_d\}|\mathbf{R}^{\mathbf{d+1}}_{=}$ is #P-hard.*

Proof. We construct a holographic reduction between $\#\{=_2\}|\{F_1, \ldots, F_{d+1}\}$ and $\#\{H_d\}|\{=_1, \ldots, =_{d+1}\}$. Let $k = d+1$.

The first problem is in domain [2], and the second problem is in domain $[m]$. The value of m will be determinated later in the construction.

We construct a matrix M of size $2 \times m$, such that for any $1 \leq i \leq k$,

$$M^{\otimes i}(=_i) = F_i. \tag{1}$$

Recall that $=_i$ and F_i denote the column vectors of length m^i and 2^i corresponding to the two functions.

Suppose there are c_j columns with the same value $\begin{pmatrix} b_j \\ b_j q_j \end{pmatrix}$ in M temporarily, $0 \leq j \leq k$. We have $M^{\otimes i}(=_i) = \sum_{j=0}^{k} c_j \begin{pmatrix} b_j \\ b_j q_j \end{pmatrix}^{\otimes i}$, because the $s_1 s_2 \cdots s_i$ columns of $M^{\otimes i}$ is $M_{s_1} \otimes \cdots \otimes M_{s_i}$ (M_s denotes the sth column of M), and the vector $=_i$ is nonzero only on entries $s_1 s_2 \cdots s_i$ satisfying $s_1 = s_2 = \cdots = s_i$.

We need that $\sum_{j=0}^{k} c_j \begin{pmatrix} b_j \\ b_j q_j \end{pmatrix}^{\otimes i} = F_i$. Notice that both sides of the equation are symmetric functions. We need $\sum_{j=0}^{k} c_j b_j^i [1, q_j, \ldots, q_j^i] = [a_0, a_1, \ldots, a_i]$ holds

for $1 \leq i \leq k$. Let $b_j = 1$, we only need $\sum_{j=0}^{k} c_j q_j^i = a_i$. Let q_j are different integers. Then, this is a system of linear equations in c_j with nonsingular Vandermonde coefficient matrix in q_j. Because q_j and a_i are integers, the solution of c_j are rational.

We look two simple cases firstly. If the solution of c_j are all nonnegative integers, it is done. If the solution of c_j are nonnegative rational numbers. Suppose p is a constant such that pc_j are nonnegative integers. Put pc_j columns $\begin{pmatrix} b_j \\ b_j q_j \end{pmatrix}$ in M. We get $M^{\otimes i}(=_i) = pF_i$. (The constant p does not change the complexity of the corresponding problem.)

In the general case, the solution of c_j may be negative rational. In fact we only need to show how to utilize b_j to realize a factor -1. Suppose we want some column $\begin{pmatrix} b \\ bq \end{pmatrix}$ in M contribute $-1 \begin{pmatrix} 1 \\ q \end{pmatrix}^{\otimes i}$ in $M^{\otimes i}(=_i) = \sum_{j=0}^{k} c_j \begin{pmatrix} b_j \\ b_j q_j \end{pmatrix}^{\otimes i}$ for $1 \leq i \leq k$.

Let $r = e^{i2\pi/(k+1)}$. Replace this column $\begin{pmatrix} b \\ bq \end{pmatrix}$ by k columns $\begin{pmatrix} b_s \\ b_s q \end{pmatrix}$, $1 \leq s \leq k$, where $b_s = r^s$. Notice

$$\sum_{s=1}^{k} b_s^i = \sum_{s=1}^{s} r^{si} = \sum_{s=0}^{k} r^{si} - 1 = \frac{1 - r^{i(k+1)}}{1 - r^i} - 1 = -1,$$

for all $1 \leq i \leq k$. These k columns indeed contribute $-1 \begin{pmatrix} 1 \\ q \end{pmatrix}^{\otimes i}$.

Suppose we get the solution of c_j from the system of linear equations. Firstly, we only take the absolute values of this solution, and handle it as the second simple case to get a matrix M. Secondly, for each c_j which should take a negative value, we replace each of its pc_j columns in M by k new columns as above to get a new M satisfying equations 1.

By theorem 2, let $H_d = (=_2)M^{\otimes 2}$, then $\#\{=_2\}|\{F_1, \ldots, F_{d+1}\}$ and $\#\{H_d\}|\mathbf{R}_{=}^{\mathbf{d+1}}$ have the same value. Obviously, the same holographic reduction exists between $\#\{=_2\}|\{F_1, \ldots, F_d\}$ and $\#\{H_d\}|\mathbf{R}_{=}^{\mathbf{d}}$.

By the results in [10], we know that $\#\{=_2\}|\{F_1, \ldots, F_d\}$ is in P, while $\#\{=_2\}|\{F_1, \ldots, F_{d+1}\}$ is #P-hard by lemma 1.

The conclusion follows from the complexity of $\#\{=_2\}|\{F_1, \ldots, F_d\}$ and $\#\{=_2\}|\{F_1, \ldots, F_{d+1}\}$. □

4 Some Partial Converse of Holographic Reduction

Since result equivalent is enough to design reductions, and algebra equivalent is a sufficient condition of result equivalent, we wonder whether it is also a necessary condition. If it is not, maybe we can explore more sufficient conditions to design new reductions. This question itself is also an interesting mathematical problem.

The converse of theorem 1 does not hold for some cases. For example $\#\{F_1 = F_2 = (1,0)\}|\{(4,1)'\}$ and $\#\{W_1 = (1,1), W_2 = (1,2)\}|\{(4,0)'\}$ always have the

same value, but obviously there is no nonsingular M such that $F_1 = W_1M$ and $F_2 = W_2M$.

Unfortunately, the converse of theorem 2 does not hold either. Consider $\#\{F_1 = F_2 = (1,0), F_3 = (1,4)\}|\{(4,1)'\}$ and $\#\{W_1 = (1,1), W_2 = (1,2), W_3 = (2,0)\}|\{(4,0)'\}$. They always give the same value. Suppose the converse of Holant theorem holds, then it must be that $F_1 = W_1M$ and $F_2 = W_2M$, so the second column of M is a zero vector. Because $F_3 = W_3M$, the second entry of F_3 should be zero. Contradiction.

It is still possible that the converse of the two theorems holds for most situation except some special cases. In the following conjecture, we simply add a condition on the arity, but maybe this is far from the right condition.

Conjecture 1. $\#\mathbf{F}|\mathbf{H}$ and $\#\mathbf{P}|\mathbf{Q}$ are two counting problems, such that at least one of $\mathbf{F}$, $\mathbf{H}$, $\mathbf{P}$, $\mathbf{Q}$ contains some function of arity more than 1. If they are result equivalent, then they are algebra equivalent.

We compare this conjecture with the following result. (The result in [17] is more general.)

Theorem 4 ([17,13]). *Suppose H_1 and H_2 are directed acyclic graphs. If for all directed acyclic graphs G, the number of homomorphisms from G to H_1 is equal to the number of homomorphisms from G to H_2, then H_1 and H_2 are isomorphic.*

Theorem 4 can be regarded as a special case of the conjecture that $\mathbf{H} = \mathbf{Q} = \mathbf{R}_=$, and $\mathbf{F} = \{H_1\}$, $\mathbf{P} = \{H_2\}$, with stronger conclusion that the matrix M in algebra equivalent is a permutation matrix.

Now we prove that if there is a matrix M keeping $=_k$ unchanged for all airty k, then it must be permutation matrix.

Suppose $M = (M_{ij})$ is $n \times n$ matrix. Let M_j denote the jth column of M, and e_j denote the standard column base vector. e_{ij} denotes the ith entry of e_j, that is, (e_{ij}) is identity matrix. We have condition $M^{\otimes k}(=_k) = (=_k)$, which means $\sum_{j=1}^{n} M_j^{\otimes k} = \sum_{j=1}^{n} e_j^{\otimes k}$, $1 \le k \le n$. Fix a i and focus on the $(ii\cdots i)$th entry of these vector equations. We get $\sum_{j=1}^{n} M_{ij}^{k} = \sum_{j=1}^{n} e_{ij}^{k}$, $1 \le k \le n$. This means $\{M_{ij}|1 \le j \le n\}$ and $\{e_{ij}|1 \le j \le n\}$ are the same multi-set, that is, each row of M is compose of 1 one and $n-1$ zeros. Suppose one column of M contains more than one 1. For example $M_{sj} = M_{tj} = 1$. Consider the $(ii\cdots ij)$th entry of these vector equations, we will get a contradiction. Hence, M is a permutation matrix.

In the rest of this paper, we prove that the conjecture holds for several classes of problems $\#\mathbf{F}|\{=_2\}$ (Holant problems). We denote this problem by $\#\mathbf{F}$ for simplicity. The range of all functions are real numbers.

Since we only consider $\#\mathbf{F}|\{=_2\}$ problems, the base M keep $=_2$ unchanged, that is, $M^{\otimes 2}(=_2) = (=_2)$. The matrix corresponds to $=_2$ is identity matrix I, so the matrix form of this equation is $MIM' = I$, which means M is orthogonal.

Theorem 5. *Suppose $\mathbf{F} = \{F_1, \ldots, F_t\}$ and $\mathbf{P} = \{P_1, \ldots, P_t\}$ are composed of unary functions over $[n]$. If $\#\mathbf{F}$ and $\#\mathbf{P}$ are result equivalent, then they are algebra equivalent.*

Proof. This a straightforward linear algebra problem.

Let F (resp. P) denote the matrix whose ith column is F_i (resp. P_i). The condition is that $F'F = P'P$.

If $t = n$ and F is full rank, then P is also full rank. Let $M = HF^{-1}$. Obviously, $MF_i = H_i$. Because $F'F = H'H = F'M'MF$, M is orthogonal matrix.

For the general case, we can show F and P has the same maximum linear independent column subset (two subset have the same element index) and the other columns are generated by this set in the same way. We can turn them into nonsingular matrices. Details omitted. □

Theorem 6. *Suppose F and H are two symmetric binary real functions over $[n]^2$. If $\#\{F\}$ and $\#\{H\}$ are result equivalent, then they are algebra equivalent.*

Proof. Suppose KFK' and LHL' are diagonal matrices and K, L are orthogonal matrices.

$\#\{F\}$ and $\#\{KFK'\}$, $\#\{H\}$ and $\#\{LHL'\}$ are algebra equivalent. We only need to prove that $\#\{KFK'\}$ and $\#\{KHK'\}$ are algebra equivalent.

Consider a cycle of length i. Since $\#\{KFK'\}$ and $\#\{LHL'\}$ have the same value on it, $\mathrm{tr}((KFK')^i) = \mathrm{tr}((LHL')^i)$.

Take $i = 1, \ldots, n$, we get that the diagonal entries sets of KFK' and LHL' are the same. Hence, $\#\{KFK'\}$ and $\#\{LHL'\}$ are algebra equivalent under a permutation matrix. □

Corollary 1. *Suppose F_1, F_2, H_1, H_2 are symmetric binary real functions over $[k]^2$, and all eigenvalues of F_1 are equal. If $\#\{F_1, F_2\}$ and $\#\{H_1, H_2\}$ are result equivalent, then they are algebra equivalent.*

Proof. By theorem 6, the eigenvalues of H_1 are also equal. If these eigenvalues are zero, then F_1 and H_1 are zero function. The conclusion holds by theorem 6.

If these eigenvalues λ are not zero, $\#\{F_1, F_2\}$ (resp. $\#\{H_1, H_2\}$) is algebra equivalent to $\#\{\lambda I, P_2\}$ (resp. $\#\{\lambda I, Q_2\}$). By theorem 6, $\#\{\lambda I, P_2\}$ and $\#\{\lambda I, Q_2\}$ are algebra equivalent. □

We need the following lemma for the next theorem.

Lemma 2. *$G = (U, V, E)$ is a bipartite graph with edge weight function $w : E \to \{1, -1\}$. If for every cycles $e_1, e_2, \ldots, e_{2k}$ of G, $w(e_1)w(e_2)\cdots w(e_{2k}) = 1$, we say G is consistent. If G is consistent, then we can extend weight function w to a complete bipartite graph $G' = (U, V, U \times V)$, such that G' is also consistent.*

Proof. Given $G = (U, V, E)$, we take a spanning tree of each of its connected components. We get a forest and denote it by $G_1 = (U, V, E_1)$. Edges in E_1 take the same weight as in G. Extend G_1 to a spanning tree $G_2 = (U, V, E_2)$, that is, $E_1 \subseteq E_2$, such that the weights of edges in $E_2 - E$ are either 1 or -1 arbitrarily. At last, we extend G_2 to complete graph G'. For each edge $(u, v) \notin E_2$, there is a unique path $P_{u,v}$ in G_2 connecting u and v. We set the weight of (u, v) to the product of weights of edges in $P_{u,v}$.

Firstly, we prove that G' is consistent. Take an arbitrary cycle C' of G'. For each edge e in C', there is a unique path P_e corresponding to it in the spanning tree G_2. We replace each edge e in C by its path P_e to get a cycle C_2 of G_2. By the definition of weights of G', the two cycles have the same product of edge weights. Each edge appears in C_2 for even many times. (Otherwise, if C_2 contains some edge e for odd many times, the cycle will start from one of the two components of the graph $(U, V, E_2 - \{e\})$ and stay in the other component.) Hence the products of edge weights are 1 for both cycles C and C'.

Secondly, we prove that G' and G give the same weight to edges in E. We prove it for all edges in each connected component of G. To reuse the notations above, we can assume that G is connected. By definition, G' and G give the same weight to edges in E_1. Because both G' and G are consistent, and the weight of $e = (u, v) \notin E_1$ is decided by the weights on the path $P_{u,v} \subseteq E_1$, they give the same weight to e. □

Theorem 7. *Suppose F_1, F_2, H_1, H_2 are symmetric binary real functions over $[k]^2$, and all eigenvalues of F_1 are different and all eigenvalues of F_2 are different. If $\#\{F_1, F_2\}$ and $\#\{H_1, H_2\}$ are result equivalent, then they are algebra equivalent.*

Proof. By theorem 6, there exist orthogonal matrices K_1, L_1 and diagonal matrix Λ_1 such that $F_1 = K_1' \Lambda_1 K_1$, $H_1 = L_1' \Lambda_1 L_1$. Because $\#\{F_1, F_2\}$ and $\#\{\Lambda_1, K_1 F_2 K_1'\}$, $\#\{H_1, H_2\}$ and $\#\{\Lambda_1, L_1 H_2 L_1'\}$, are algebra equivalent, $\#\{\Lambda_1, K_1 F_2 K_1'\}$ and $\#\{\Lambda_1, L_1 H_2 L_1'\}$ are result equivalent.

To prove the conclusion, we only need to prove $\#\{\Lambda_1, K_1 F_2 K_1'\}$ and $\#\{\Lambda_1, L_1 H_2 L_1'\}$ are algebra equivalent. We use F (resp. H) to denote $K_1 F_2 K_1'$ (resp. $L_1 H_2 L_1'$).

Applying theorem 6 to $\#\{F\}$ and $\#\{H\}$, there exist orthogonal matrices K, L and diagonal matrix Λ such that $F = K \Lambda K'$, $H = L \Lambda L'$. The diagonal entries of Λ are different eigenvalues.

Let σ_i denote the binary relation $\{(i, i)\}$, $i \in [n]$. Consider a circle of $r + s$ with r functions Λ_1 and s functions F. The value of $\#\{\Lambda_1, F\}$ on this instance is $tr(\Lambda_1^r F^s) = tr(\Lambda_1^r K \Lambda^s K')$. This equation holds for any r and s. Suppose the ith diagonal entry of Λ_1 is λ_i. $tr(\Lambda_1^r K \Lambda^s K') = \Sigma_i \lambda_i^r tr(\sigma_i K \Lambda^s K')$. Fix s and take n different values of r. We get a nonsingular system of linear equations in $tr(\sigma_i K \Lambda^s K')$, with Vandermonde coefficient matrix in λ_i.

Similar analysis also holds for $tr(\Lambda_1^r L \Lambda^s L')$. By the conditions $tr(\Lambda_1^r F^s) = tr(\Lambda_1^r H^s)$. Two systems of linear equations are the same, so if $\lambda_i \neq 0$, $tr(\sigma_i K \Lambda^s K') = tr(\sigma_i L \Lambda^s L')$. If there is some unique $\lambda_{i_0} = 0$, notice $tr((\Sigma_{i \in [n]} \sigma_i) K \Lambda^s K') = tr(=_2 K \Lambda^s K') = tr(L \Lambda^s L') = tr((\Sigma_{i \in [n]} \sigma_i) L \Lambda^s L')$, we also have $tr(\sigma_{i_0} K \Lambda^s K') = tr(\sigma_{i_0} L \Lambda^s L')$.

Similar analysis holds for Λ part. Hence, for any i, j, $tr(\sigma_i K \sigma_j K') = tr(\sigma_i L \sigma_j L')$, which means $(K(i, j))^2 = (L(i, j))^2$.

Define a weighted undirected bipartite graph $G = (U, V, E)$, such that $(i, j) \in E$ iff $L(i, j) \neq 0$. The weight of (i, j) is $w((i, j)) = K(i, j)/L(i, j) \in \{1, -1\}$.

We just consider $\Lambda_1^r F^s$ in the above analysis. If consider $\Lambda_1^{r_1} F^{r_2} \Lambda_1^{r_3} F^{r_4}$, we can get for any $i, j, k, l \in [n]$, $tr(\sigma_i K \sigma_j K' \sigma_k K \sigma_l K') = tr(\sigma_i L \sigma_j L' \sigma_k L \sigma_l L')$,

which means $K(i,j)K(k,j)K(k,l)K(i,l) = L(i,j)L(k,j)L(k,l)L(i,l)$. If none of them is zero, $((i,j),(k,j),(k,l),(i,l))$ is a cycle in G, and the equation says G is consistent on this cycle (the product of edge weights in the cycle is 1). If we consider arbitrary alternations between $\mathit{\Lambda}$ and F, we get arbitrary cycles, so G is consistent. By lemma 2, we can get a consistent complete bipartite graph G'. Suppose the weight function of G' is $W(i,j)$, which is also a matrix.

For any 2×2 submatrix $W(\{i,k\},\{j,l\})$ of W, its two rows are identical or linear dependent by a factor -1 (consider the circle $((i,j),(k,j),(k,l),(i,l))$ in G'). Hence the rank of W is 1.

Suppose W is the product of two ± 1 valued vectors $\delta_1 \delta_2'$. Let $\mathrm{diag}(\delta_\mathrm{i})$ denote the diagonal matrix whose diagonal is δ_i. Since $W(i,j)L(i,j) = K(i,j)$ for all entries, $\mathrm{diag}(\delta_1)$ L $\mathrm{diag}(\delta_2) = \mathrm{K}$. Notice $\mathrm{diag}(\delta_1)\ \Lambda_1\ \mathrm{diag}(\delta_1) = \Lambda_1$ and $\mathrm{diag}(\delta_2)\ \Lambda\ \mathrm{diag}(\delta_2) = \Lambda$. $\#\{\mathit{\Lambda}_1, F\}$ and $\#\{\mathit{\Lambda}_1, H\}$ are algebra equivalent by matrix $\mathrm{diag}(\delta_1)$. □

Corollary 2. *Suppose F_1, F_2, H_1, H_2 are symmetric binary real functions over $[2]^2$. If $\#\{F_1, F_2\}$ and $\#\{H_1, H_2\}$ are result equivalent, then they are algebra equivalent.*

Proof. Since the size of domain is 2, either one of F_1 and F_2 have the same eigenvalues, or neither of them has.

The first case is by corollary 1, the second case is by theorem 7. □

Corollary 3. *Suppose F_1, H_1 are unary real functions over $[2]$, and F_2, H_2 are symmetric binary real functions over $[2]^2$. If $\#\{F_1, F_2\}$ and $\#\{H_1, H_2\}$ are result equivalent, then they are algebra equivalent.*

Proof. Let binary functions $F_3 = F_1^{\otimes 2}$, $H_3 = H_1^{\otimes 2}$. Apply corollary 2 to $\#\{F_3, F_2\}$ and $\#\{H_3, H_2\}$.

There exists orthogonal matrix M, such that $M^{\otimes 2}F_3 = H_3$, $M^{\otimes 2}F_2 = H_2$

Because $M^{\otimes 2}F_3 = H_3$, which means $(MF_1)^{\otimes 2} = H_1^{\otimes 2}$, either $MF_1 = H_1$ or $MF_1 = -H_1$. If $MF_1 = H_1$, the conclusion holds by base M. If $MF_1 = -H_1$, the conclusion holds by base $-M$. □

Theorem 8. *Suppose F_3, H_3 are symmetric ternary real functions over $[2]^3$, If $\#\{F_3\}$ and $\#\{H_3\}$ are result equivalent, then they are algebra equivalent.*

Proof. Suppose $H_3 = [h_0, h_1, h_2, h_3]$. F_1 (resp. H_1) denote the unary function by connecting two inputs of F_3 (resp. H_3) using $=_2$, that is, $H_1 = [h_0 + h_2, h_1 + h_3]$.

Let F_2 (resp. H_2) denote the binary function which is one F_3 (resp. H_3) function connected by one F_1 (resp. H_1).

Obviously, $\#\{F_1, F_2\}$ and $\#\{H_1, H_2\}$ are result equivalent. Apply corollary 3 to $\#\{F_1, F_2\}$ and $\#\{H_1, H_2\}$. There exists orthogonal matrix M such that $MF_1 = H_1$ and $M^{\otimes 2}F_2 = H_2$.

Let $M^{\otimes 3}F_3 = [f_0, f_1, f_2, f_3]$, $\mathit{\Delta}_j = h_j - f_j$, $j = 0, 1, 2, 3$. Because $MF_1 = [f_0 + f_2, f_1 + f_3]$ and $MF_1 = H_1$, $\mathit{\Delta}_0 + \mathit{\Delta}_2 = 0$ and $\mathit{\Delta}_1 + \mathit{\Delta}_3 = 0$.

Let $H_1 = [h_0 + h_2, h_1 + h_3] = MF_1 = [a, b]$, then $H_2 = [ah_0 + bh_1, ah_1 + bh_2, ah_2 + bh_3]$ and $M^{\otimes 2}F_2 = [af_0 + bf_1, af_1 + bf_2, af_2 + bf_3]$. ($F_2$ is composed

of F_3 and F_1. Because M is orthogonal, $M^{\otimes 2}F_2$ can be realized by composing $M^{\otimes 3}F_3$ and MF_1.) Because $M^{\otimes 2}F_2 = H_2$, $a\Delta_0 + b\Delta_1 = 0$, $a\Delta_1 + b\Delta_2 = 0$, $a\Delta_2 + b\Delta_3 = 0$.

Now we get five linear equations about Δ_j. Notice a and b are real numbers, since F_3 and H_3 are real functions. Calculation shows that, there is only zero solution iff $a^2 + b^2 \neq 0$. Hence, if $a \neq 0$ or $b \neq 0$, we have proved $M^{\otimes 3}F_3 = H_3$.

If $a = b = 0$, $H_1 = MF_1 = [0, 0]$, so $F_1 = [0, 0]$. let $F_3 = [x, y, -x, -y]$. There exist $s = (x - iy)/2$ and $t = (x + iy)/2$ such that $F_3 = s\begin{pmatrix}1\\ i\end{pmatrix}^{\otimes 3} + t\begin{pmatrix}1\\ -i\end{pmatrix}^{\otimes 3}$. Because x and y are real number, $s \neq 0$, $t \neq 0$. Under base $\begin{pmatrix}1 & 1\\ i & -i\end{pmatrix}$, $\#\{F_3\}|\{F_3\}$ is holographic reduced to $\#\{[s, 0, 0, t]\}|\{[8t, 0, 0, 8s]\}$. Similarly, H_3 also has form $H_3 = c\begin{pmatrix}1\\ i\end{pmatrix}^{\otimes 3} + d\begin{pmatrix}1\\ -i\end{pmatrix}^{\otimes 3}$, and under the same base, $\#\{H_3\}|\{H_3\}$ is holographic reduced to $\#\{[c, 0, 0, d]\}|\{[8d, 0, 0, 8c]\}$.

Because $\#\{F_3\}|\{F_3\}$ is special case of $\#\{F_3\}$, $\#\{[s, 0, 0, t]\}|\{[8t, 0, 0, 8s]\}$ and $\#\{[c, 0, 0, d]\}|\{[8d, 0, 0, 8c]\}$ are result equivalent. Hence, $st = cd$.

Notice $\begin{pmatrix}(\frac{c}{s})^{\frac{1}{3}} & 0\\ 0 & (\frac{d}{t})^{\frac{1}{3}}\end{pmatrix}$ can turn $[s, 0, 0, t]$ into $[c, 0, 0, d]$. Let

$$M = \begin{pmatrix}1 & 1\\ i & -i\end{pmatrix}\begin{pmatrix}(\frac{c}{s})^{\frac{1}{3}} & 0\\ 0 & (\frac{d}{t})^{\frac{1}{3}}\end{pmatrix}\begin{pmatrix}1 & 1\\ i & -i\end{pmatrix}^{-1}.$$

Then, $M^{\otimes 3}F_3 = H_3$ and $M'M = I$ (the second equation is right because of the condition $st = cd$). □

References

1. Bulatov, A., Grohe, M.: The complexity of partition functions. Theor. Comput. Sci. 348(2-3), 148–186 (2005)
2. Bulatov, A.: The complexity of the counting constraint satisfaction problem. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 646–661. Springer, Heidelberg (2008)
3. Bulatov, A., Dyer, M., Goldberg, L., Jalsenius, M., Richerby, D.: The complexity of weighted Boolean #CSP with mixed signs. Theor. Comput. Sci. 410(38-40), 3949–3961 (2009)
4. Cai, J.-Y., Choudhary, V.: Valiant's holant theorem and matchgate tensors. Theor. Comput. Sci. 384(1), 22–32 (2007)
5. Cai, J.-Y., Lu, P.: Holographic algorithms: The power of dimensionality resolved. Theor. Comput. Sci. 410(18), 1618–1628 (2009)
6. Cai, J.-Y., Lu, P.: On blockwise symmetric signatures for matchgates. Theor. Comput. Sci. 411(4-5), 739–750 (2010)
7. Cai, J.-Y., Lu, P.: Holographic algorithms: from art to science. In: STOC 2007, pp. 401–410 (2007)
8. Cai, J.-Y., Chen, X., Lu, P.: Graph homomorphisms with complex values: a dichotomy theorem. CoRR abs/0903.4728 (2009)

9. Cai, J.-Y., Lu, P., Xia, M.: Holographic algorithms by Fibonacci gates and holographic reductions for hardness. In: FOCS 2008, pp. 644–653 (2008)
10. Cai, J.-Y., Lu, P., Xia, M.: Holant problems and counting CSP. In: STOC 2009, pp. 715–724 (2009)
11. Cai, J.-Y., Lu, P., Xia, M.: A computational proof of complexity of some restricted counting problems. In: TAMC 2009, pp. 138–149 (2009)
12. Creignou, N., Hermann, M.: Complexity of generalized satisfiability counting problems. Inf. Comput. 125(1), 1–12 (1996)
13. Dyer, M., Goldberg, L., Paterson, M.: On counting homomorphisms to directed acyclic graphs. J. ACM 54(6) (2007)
14. Dyer, M., Goldberg, L., Jerrum, M.: The complexity of weighted Boolean CSP. SIAM J. Comput. 38(5), 1970–1986 (2009)
15. Dyer, M., Goldberg, L., Jerrum, M.: A complexity dichotomy for hypergraph partition functions. CoRR abs/0811.0037 (2008)
16. Dyer, M., Greenhill, C.: The complexity of counting graph homomorphisms. Random Struct. Algorithms 17(3-4), 260–289 (2000)
17. Lovász, L.: Operations with structures. Acta Math. Acad. Sci. Hung. 18, 321–328 (1967)
18. Vadhan, S.: The complexity of counting in sparse, regular, and planar graphs. SIAM J. Comput. 31(2), 398–427 (2001)
19. Valiant, L.: The complexity of computing the permanent. Theor. Comput. Sci. 8, 189–201 (1979)
20. Valiant, L.: The complexity of enumeration and reliability problems. SIAM J. Comput. 8(3), 410–421 (1979)
21. Valiant, L.: Holographic algorithms. SIAM J. Comput. 37(5), 1565–1594 (2008)
22. Valiant, L.: Holographic circuits. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1–15. Springer, Heidelberg (2005)
23. Valiant, L.: Accidental algorthims. In: FOCS 2006, pp. 509–517 (2006)
24. Xia, M., Zhang, P., Zhao, W.: Computational complexity of counting problems on 3-regular planar graphs. Theor. Comput. Sci. 384(1), 111–125 (2007)

Optimal Trade-Offs for Succinct String Indexes

Roberto Grossi[1], Alessio Orlandi[1], and Rajeev Raman[2]

[1] Dipartimento di Informatica, Università di Pisa
{grossi,aorlandi}@di.unipi.it
[2] Department of Computer Science, University of Leicester, United Kingdom
r.raman@mcs.le.ac.uk

Abstract. Let s be a string whose symbols are solely available through $\texttt{access}(i)$, a read-only operation that *probes* s and returns the symbol at position i in s. Many compressed data structures for strings, trees, and graphs, require two kinds of queries on s: $\texttt{select}(c, j)$, returning the position in s containing the jth occurrence of c, and $\texttt{rank}(c, p)$, counting how many occurrences of c are found in the first p positions of s. We give matching upper and lower bounds for this problem. The main contribution is to introduce a general technique for proving lower bounds on succinct data structures, that is based on the access patterns of the supported operations, abstracting from the particular operations at hand.

1 Introduction

We are given a read-only sequence $s \equiv s\,[0, n-1]$ of n symbols over an integer alphabet $\Sigma = [\sigma] \equiv \{0, 1, \ldots, \sigma - 1\}$, where $2 \leq \sigma \leq n$. The symbols in s can be read using $\texttt{access}(i)$, for $0 \leq i \leq n-1$: this primitive *probes* s and returns the symbol at position i, denoted by $s\,[i]$. Given the sequence s, its length n, and the alphabet size σ, we want to support the following query operations for a symbol $c \in \Sigma$: $\texttt{select}(c, j)$ returns the position inside s containing the jth occurrence of symbol c, or -1 if that occurrence does not exist; $\texttt{rank}(c, p)$ counts how many occurrences of c are found in $s\,[0, p-1]$.

We postulate that an auxiliary data structure, called a *succinct index*, is constructed in a preprocessing step to help answer these queries rapidly. In this paper, we study the natural and fundamental *time-space* tradeoff between two parameters t and r for this problem:

- $t =$ the *probe complexity*, which is the maximal number of probes to s (i.e. calls to $\texttt{access}$) that the succinct index makes when answering a query[1];
- $r =$ the *redundancy*, which is the number of bits required by the succinct index, and does *not* include the space needed to represent s itself.

Clearly, these queries can be answered in negligible space but $O(n)$ probes by scanning s, or in zero probes by making a copy of s in auxiliary memory at

[1] The time complexity of our results in the RAM model with logarithmic-sized words is linearly proportional to the probe complexity. Hence, we focus on the latter.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 678–689, 2010.

preprocessing time, but with redundancy of $\Theta(n \log \sigma)$ bits. We are interested in succinct indices that use few probes, and have redundancy $o(n \log \sigma)$, i.e., asymptotically smaller than the space for s itself. Specifically, we obtain upper and lower bounds on the redundancy $r \equiv r(t, n, \sigma)$, viewed as a function of the maximum number t of probes, the length n of s, and the alphabet size σ. We assume that $t > 0$ in the rest of the paper.

Motivation. Succinct indices have numerous applications to problems involving indexing massive data sets [1]. The `rank` and `select` operations are basic primitives at the heart of many sophisticated indexing data structures for strings, trees, graphs, and sets [10]. Their efficiency is crucial to make these indexes fast and space-economical. Our results are most interesting for the case of "large" alphabets, where σ is a not-too-slowly growing function of n. Large alphabets are common in modern applications: e.g. many files are in Unicode character sets, where σ is of the order of hundreds or thousands. Inverted lists or documents in information retrieval systems can be seen as sequences s of words, where the alphabet Σ is obviously large and increasing with the size of the collection (it is the vocabulary of distinct words appearing over the entire document repository).

Our results. Our first contribution is showing that the redundancy r in bits

$$r(t, n, \sigma) = \Theta\left(\frac{n \log \sigma}{t}\right) \tag{1}$$

is tight for any succinct index solving our problem, for $t = O(\log \sigma / \log \log \sigma)$. (All the logarithms in this paper are to the base 2.) We provide matching upper and lower bounds for this range of values on t, under the assumption that $O(t)$ probes are allowed for `rank` and `select`, i.e. we ignore multiplicative constant factors. The result is composed by a lower bound of $r = \Omega(\frac{n \log \sigma}{t})$ bits that holds for $t = o(\log \sigma)$ and by an upper bound of $r = O(\frac{n \log \sigma}{t} + n \log \log \sigma)$. We will also provide a lower bound of $r = \Omega(\frac{n \log t}{t})$ for $t = O(n)$ in the full version, extending the $\sigma = 2$ case of [7]. This leaves open what is the optimal redundancy when $t = \Omega(\frac{\log \sigma}{\log \log \sigma})$. Running times for the upper bound are $O(t + \log \log \sigma)$ for `rank` and $O(t)$ for `select`.

An interpretation of (1) is that, given a data collection D, if we want to build an additional succinct index on D that saves space by a factor t over that taken by D, we have to pay $\Omega(t)$ access cost for the supported queries. Note that the plain storage of the sequence s itself requires $n \log \sigma$ bits. Moreover, our result shows that it is suboptimal to build σ individual succinct indexes (like those for the binary-alphabet case, e.g. [16]), one per symbol $c \in [\sigma]$: the latter approach has redundancy $\Theta(\frac{\sigma n \log t}{t})$ while the optimal redundancy is given in eq. (1), when $t = O(\log \sigma / \log \log \sigma)$.

Lower bounds are our main findings, while the matching upper bounds are derived from known algorithmic techniques. Thus, our second contribution is a general technique that extends the algorithmic encoding/decoding approach in [4] in the sense that it abstracts from the specific query operation at hand, and focuses on its access pattern solely. For this, we can single out a sufficiently large,

conflict free subset of the queries that are classified as *stumbling* or *z-unique*. In the former case, we extract direct knowledge from the *probed* locations; in the latter, the novelty of our approach is that we can extract (implicit) knowledge also from the *unprobed* locations. We are careful not to exploit the specific semantics of the query operations at this stage. As a result, our technique applies to other kinds of query operations for predecessor, prefix sum, permutation, and pattern searching problems, to name a few, as long as we can extract a sufficiently large subset of the queries with the aforementioned features. We will discuss them extensively in the full version.

We also provide further running times for the rank/select problem. For example, if $\sigma = (\log n)^{O(1)}$, the `rank` operation requires only $O(t)$ time; also, we can get $O(t \log\log\sigma \log^{(3)}\sigma)$ time[2] for `rank` and $O(t \log\log\sigma)$ time for `select` (Theorem 4). As a corollary, we can obtain an entropy-compressed data structure that represents s using $nH_k(s) + O(\frac{n \log\sigma}{\log\log\sigma})$ bits, for any $k = o(\frac{\log_\sigma n}{\log\log\sigma})$, supporting `access` in $O(1)$ time, `rank` and `select` in $O(\log\log\sigma)$ time (here, $H_k(s)$ is the kth-order empirical entropy).

Related work. The conceptual separation of the index from the input data was introduced to prove lower bounds in [6,13]. It was then explicitly employed for upper bounds in [5,11,17], and was fully formalized in [1]. The latter contains the best known upper bounds for our problem[3], i.e. $O(s)$ probes for `select` and $O(s \log k)$ probes for `rank`, for any two parameters $s \leq \log\sigma/\log\log\sigma$ and $k \leq \sigma$, with redundancy $O(n \log k + n(1/s + 1/k)\log\sigma)$. For example, fixing $s = k = \log\log\sigma$, they obtain $O(\log\log\sigma)$ probes for `select` and $O(\log\log\sigma \log^{(3)}\sigma)$ probes for `rank`, with redundancy $O(n \log\sigma/\log\log\sigma)$. By eq. (1), we get the same redundancy with $t = O(\log\log\sigma)$ probes for both `rank` and `select`. Hence, our probe complexity for `rank` is usually better than [1] while that of `select` is the same. Our $O(\log\log\sigma)$ running times are all better when compared to $O((\log\log\sigma)^2 \log^{(3)}\sigma)$ for `rank` and $O((\log\log\sigma)^2)$ for `select` in [1].

2 General Technique

This section aims at stating a general lower bound technique, of independent interest, which applies not only to both `rank` and `select` but to other query operations as well. Suppose we have a set S of strings of length n, and a set Q of queries that to be supported on S using at most t probes each and an unknown redundancy r . Under certain assumptions on S and Q, we can show a lower bound on r. Clearly, any choice of S and Q is allowed for the upper bound.

Terminology. We now give a framework that relies on a simple notion of entropy $H(S)$, where $H(X) = \lceil \log |X| \rceil$ for any class of $|X|$ combinatorial objects [3]. The framework extends the algorithmic encoding/decoding approach [4]. Consider an arbitrary algorithm $\mathcal{A}$ that can answer to any query in Q performing at most

[2] We define $\log^{(1)} x := \log_2 x$ and for integer $i \geq 2$, $\log^{(i)} x := \log_2(\log^{(i-1)} x)$.

[3] We compare ourselves with the improved bounds given in the full version of [1].

t probes on any $s \in S$, using a succinct index with r bits. We describe how to encode s using $\mathcal{A}$ and the succinct index as a black box, thus obtaining $E(s)$ bits of encoding. Then, we describe a decoder that knowing $\mathcal{A}$, the index of r bits, and the latter encoding of $E(s)$ bits, is able to reconstruct s in its original form. The encoding and decoding procedure are allowed unlimited (but finite) computing time, alibet $\mathcal{A}$ can make at most t probes per query.

The lower bound on r arises from the necessary condition $\max_{s \in S} E(s) + r \geq H(S)$, since otherwise the decoder cannot be correct. Namely, $r \geq H(S) - \max_s E(s)$: the lower $E(s)$, the tighter the lower bound for r. Our contribution is to give conditions on S and Q so that the above approach can hold for a variety of query operations, and is mostly oblivious of the specific operation at hand since the query access pattern to s is relevant. This appears to be novel.

First, we require S to be sufficiently dense, that is, $H(S) \geq n \log \sigma - \Theta(n)$. Second, Q must be a subset of $[\sigma] \times [n]$, so that the first parameter specifies a character c and the second one an integer p. Elements of Q are written as $q_{c,p}$. Third, answers to queries must be within $[n]$. The set Q must contain a number of stumbling or z-unique queries, as we define now. Consider an execution of $\mathcal{A}$ on a query $q_{c,p} \in Q$ for a string s. The set of accessed position in s, expressed as a subset of $[n]$ is called an *access pattern*, and is denoted by $\mathrm{Pat}_s(q_{c,p})$.

First, *stumbling* queries imply the occurrence of a certain symbol c inside their own access pattern: its position can be decoded by using just the answer and the parameters of the query. Formally, $q_{c,p} \in Q$ is stumbling if there exists a computable function f that takes in input c, p and the answer of $q_{c,p}$ over s, and outputs a position $x \in \mathrm{Pat}_s(q_{c,p})$ such that $s[x] = c$; x is called the *target* of $q_{c,p}$. The rationale is that the encoder can avoid storing any information regarding $s[x] = c$, since x can be extracted by the decoder from f and the t probed positions by $\mathcal{A}$. We let $Q'_s \subseteq Q$ be the set of stumbling queries over s.

Second, *z-unique* queries are at the heart of our technique, where z is a positive integer. Informally, they have specific answers implying unique occurrences of a certain symbol c in a segment of s of length $z + 1$. Formally, a set U of answers is *z-unique* if for any $q_{c,p}$ having answer in U, there exists a *unique* $i \in [p, p + z]$ such that $s[i] = c$ (i.e. $s[j] \neq c$ for all $j \in [p, p + z], j \neq i$). A query $q_{c,p}$ having answer in U is called z-unique and the corresponding position i is called the *target* of $q_{c,p}$. Note that, to our purposes, we will restrict to the cases where $H(U) = O(n)$. The rationale is the following: when the decoder wants to rebuild the string it must generate queries, execute them, and test whether they are z-unique by checking if their answers are in U. Once that happens, it can infer a position i such that $s[i] = c$, even though such a position is not probed by the query. We denote by $Q''_s(z) \subseteq Q \setminus Q'_s$ the set of z-unique queries over s that are *not* stumbling. We also let $\mathrm{Tgt}_s(q_{c,p})$ denote the target of query $q_{c,p}$ over s, if it exists, and let $\mathrm{Tgt}_s(Q) = \cup_{q \in Q} Tgt_s(q)$ for any set of queries Q.

Main statement. We now state our main theorem. Let S be a set of strings such that $H(S) \geq n \log \sigma - \Theta(n)$. Consider a set of queries Q that can be answered by performing at most t probes per query and using r bits of redundancy.

Theorem 1. *For any $z \in [\sigma]$, let $\lambda(z) = \min_{s \in S} |Tgt_s(Q'_s) \cup Tgt_s(Q''_s(z))|$. Then, there exists integers γ and δ with $\min\{\lambda(z), n\}/(15t) \leq \gamma + \delta \leq \lambda(z)$, such that any succinct index has redundancy*

$$r \geq \gamma \log\left(\frac{\sigma}{z}\right) + \delta \log\left(\frac{\sigma\delta}{t|Q|}\right) - \Theta(n).$$

The proof goes through a number of steps, each dealing with a different issue and is deferred to Section 3.

Applications. We now apply Theorem 1 to our two main problems, for an alphabet size $\sigma \leq n$.

Theorem 2. *Any algorithm solving* `rank` *queries on a string $s \in [\sigma]^n$ using at most $t = o(\log \sigma)$ character probes (i.e.* `access` *queries), requires a succinct index with $r = \Omega\big(\frac{n \log \sigma}{t}\big)$ bits of redundancy.*

Proof. We start by defining the set S of strings. For the sake of presentation, suppose σ divides n. An arbitrary string $s \in S$ is the concatenation of n/σ *permutations* of $[\sigma]$. Note that $|S| = (\sigma!)^{n/\sigma}$ and so we have $H(S) \geq n \log \sigma - \Theta(n)$ bits (by Stirling's approximation).

Without loss of generality, we prove the bound on a derivation of the `rank` problem. We define the set Q and fix the parameter $z = \sigma^{3/4}\sqrt{t}$, so that the queries are $q_{c,p} = \texttt{rank}(c, p + z) - \texttt{rank}(c, p)$, where $c \in [\sigma]$ and $p \in [n]$ with $p \bmod z \equiv 0$. In this setting, the z-unique answers are in $U = \{1\}$. Indeed, whenever $q_{c,p} = 1$, there exists just one instance of c in $s[p, p + z]$. Note that $|Q| = n\sigma/z > n$, for σ larger than some constant.

Observe that $\lambda(z) \geq n$, as each position i in s such that $s[i] = c$, is the target of exactly one query $q_{c,p}$: supposing the query is not stumbling, such a query is surely z-unique. By Theorem 1, $\gamma + \delta \geq n/(30t)$ since a single query is allowed to make up to $2t$ probes now. (This causes just a constant multiplicative factor in the lower bound.) Having met all requirements, we apply Theorem 1, and get $r \geq F$, where $F := \gamma \log\left(\frac{\sigma}{z}\right) - \delta \log\big(\frac{nt}{z\delta}\big)$.

We distinguish between two cases. If $\delta \leq n/\sigma^{1/4}$, then $\delta \log((nt)/(z\delta)) \leq (n/\sigma^{1/4}) \log\big((nt\sigma^{1/4})/(nz)\big) \leq (n/2\sigma^{1/4}) \log(t/\sigma)$, since $\delta \log(1/\delta)$ is monotone increasing in δ as long as $\delta \leq \lambda(z)/2$ (and $n/\sigma^{1/4} \leq \lambda(z)/2$ for sufficiently large σ). Hence, recalling that $t = o(\log \sigma)$, the absolute value of the second term in F is $o(n/t)$ for σ larger than a constant. Moreover, $\gamma \geq n/(30t) - \delta \geq n/(60t)$ in this setting, so that the bound $r \geq F$ reduces to $r \geq (n/240t) \log \sigma - (n/120t) \log t - \Theta(n) = (n/240t) \log \sigma - \Theta(n)$. In the other case, we have $\delta \geq n/\sigma^{1/4}$, and $\delta \log(\frac{nt}{z\delta}) \leq \delta \log\big((\sigma^{1/4}t)/(\sigma^{3/4}\sqrt{t})\big) = (\delta/2) \log(t/\sigma)$. Therefore, we know that $\gamma \log(\sigma/z) + (\delta/2) \log(\sigma/t) \geq \frac{1}{2}(\gamma + \delta) \log(\sigma/z)$, as we chose $z \geq t$. Again, we obtain $r \geq (n/120t) \log \sigma - \Theta(n)$. In both cases, the $\Theta(n)$ term is negligible as $t = o(\log \sigma)$, hence the bound. □

Theorem 3. *Any algorithm solving* `select` *queries on a string $s \in [\sigma]^n$ using at most $t = o(\log \sigma)$ character probes (i.e.* `access` *queries), requires a succinct index with $r = \Omega\big(\frac{n \log \sigma}{t}\big)$ bits of redundancy.*

Proof. The set S of strings is composed by *full strings*, assuming that σ divides n. A full string contains each character exactly n/σ times and, differently from Theorem 2, has no restrictions on where they can be found. Again, we have $H(S) \geq n \log \sigma - \Theta(n)$.

The set Q of queries is $q_{c,p} = \texttt{select}(c, p)$, where $p \in [n/\sigma]$, and all queries in Q are stumbling ones, as $\texttt{select}(c, i) = x$ immediately implies that $s[x] = c$ (so f is the identity function). There are no z-unique queries here, so we can fix any value of z: we choose $z = 1$. It is immediate to see that $\lambda(z) = n$, and $|Q| = n$, as there are only n/σ queries for each symbols in $[\sigma]$. By Theorem 1, we know that $\gamma + \delta \geq n/(15t)$. Hence, the bound is $r \geq \gamma \log \sigma + \delta \log\left(\frac{\sigma\delta}{nt}\right) \geq (n/15t) \log(\sigma/t^2) - \Theta(n)$. Again, as $t = o(\log \sigma)$ the latter term is negligible. □

3 Proof of Theorem 1

We give an upper bound on $E(s)$ for any $s \in S$ by describing an encoder and a decoder for s. In this way we can use the relation $\max_{s \in S} E(s) + r \geq H(S)$ to induce the claimed lower bound on r (see Sec. 2). We start by exploiting z-unique and stumbling queries to encode a single position and its content compactly. Next, we deal with conflicts between queries: not all queries in Q are useful for encoding. We describe a mechanical way to select a sufficiently large subset of Q so that conflicts are avoided. Bounds on γ and λ arise from such a process. Finally, we show how to store query parameters to be read by the decoder.

Entropy of a single position and its content. We first evaluate the entropy of positions and their contents by exploiting the knowledge of z-unique and stumbling queries. We use the notation $H(S|\Omega)$ for some event Ω as a shortcut for $H(S')$ where $S' = \{s \in S | s \text{ satisfies } \Omega\}$.

Lemma 1. *For any $z \in [\sigma]$, let $\Omega_{c,p}$ be the condition "$q_{c,p}$ is z-unique". Then it holds $H(S) - H(S|\Omega_{c,p}) \geq \log(\sigma/z) - O(1)$.*

Proof. Note that set $(S|\Omega_{c,p}) = \{s \in S : \mathrm{Tgt}_s(q_{c,p}) \text{ is defined on } s\}$ for a given query $q_{c,p}$. It is $|(S|\Omega_{c,p})| \leq (z+1)\sigma^{n-1}$ since there at most $z+1$ candidate target cells compatible with $\Omega_{c,p}$ and at most $|S|/\sigma$ possible strings with position containing c at a fixed position. So, $H(S|\Omega_{c,p}) \leq \log(z+1) + H(S) - \log \sigma$. □

Lemma 2. *Let $\Omega'_{c,p}$ be the condition "$q_{c,p}$ is a stumbling query". Then, it holds that $H(S) - H(S|\Omega'_{c,p}) \geq \log(\sigma/t) - O(1)$.*

Proof. The proof for this situation is already known from [8]. In our notation, the proof goes along the same lines as that of Lemma 1, except that we have t choices instead of $z+1$. To see that, let $m_1, m_2, \ldots, m_t$ be the positions, in temporal order, probed by the algorithm $\mathcal{A}$ on s while answering $q_{c,p}$. Since the query is stumbling, the target will be one of $m_1, \ldots, m_t$. It suffices to remember which one of the t steps probe that target, since their values $m_1, \ldots, m_t$ are deterministically characterized given $\mathcal{A}$, s, $q_{c,p}$. □

Conflict handling. In general, multiple instances of Lemma 1 and/or Lemma 2 cannot be applied independently. We introduce the notion of conflict on the targets and show how to circumvent this difficulty. Two queries $q_{b,o}$ and $q_{c,p}$ *conflict* on s if at least one of the following three condition holds: (i) $\mathrm{Tgt}_s(q_{c,p}) \in \mathrm{Pat}_s(q_{b,o})$, (ii) $\mathrm{Tgt}_s(q_{b,o}) \in \mathrm{Pat}_s(q_{c,p})$, (iii) $\mathrm{Tgt}_s(q_{c,p}) = \mathrm{Tgt}_s(q_{b,o})$. A set of queries where no one conflicts with another is called *conflict free*. The next lemma is similar to the one found in [9], but the context is different.

Lemma 3 defines a lower bound on the maximum size of a conflict free subset of Q. We use an iterative procedure that maintains at each ith step a set Q_i^* of conflict free queries and a set C_i of available targets, such that no query q whose target is in C_i will conflict with any query $q' \in Q_{i-1}^*$. Initially, C_0 contains all targets for the string s, so that by definition $|C_0| \geq \lambda(z)$. Also, Q_0^* is empty.

Lemma 3. *Let $i \geq 1$ be an arbitrary step and assume $|C_{i-1}| > 2|C_0|/3$. Then, there exists Q_i^* and C_i such that (a) $|Q_i^*| = 1 + |Q_{i-1}^*|$, (b) Q_i^* is conflict free, (c) $|C_i| \geq |C_0| - 5it \geq \lambda(z) - 5it$.*

By applying Lemma 3 until $|C_i| \leq 2|C_0|/3$, we obtain a final set Q^*, hence:

Corollary 1. *For any $s \in S$, $z \in [\sigma]$, there exists a set Q^* containing z-unique and stumbling queries of size $\gamma + \delta \geq \min\{\lambda(z), n\}/(15t)$, where $\gamma = |\{q \in Q^* | q \text{ is stumbling on } s\}|$ and $\delta = |\{q \in Q^* | q \text{ is } z\text{-unique on } s\}|$.*

Encoding. We are left with the main task of describing the encoder. Ideally, we would like to encode the targets, each with a cost as stated in Lemma 1 and Lemma 2, for the conflict free set Q^* mentioned in Corollary 1. Characters in the remaining positions can be encoded naively as a string. This approach has a drawback. While encoding which queries in Q are stumbling has a payoff when compared to Lemma 2, we don't have such a guarantee for z-unique queries when compared to Lemma 1. Without getting into details, according to the choice of the parameters $|Q|$, z and t, such encoding sometimes saves space and sometimes does not: it may use even more space than $H(S)$. For example, when $|Q| = O(n)$, even the naive approach works and yields an effective lower bound. Instead, if Q is much larger, savings are not guaranteed. The main point here is that we want to overcome such a dependence on the parameters and always guarantee a saving, which we obtain by means of an implicit encoding of z-unique queries.

Archetype and trace. Instead of trying to directly encode the information of Q^* as discussed above, we find a query set Q^A called the *archetype* of Q^*, that is indistinguishable from Q^* in terms of γ and δ. The extra property of Q^A is to be decodable using just $O(n)$ additional bits, hence $E(s)$ is smaller when Q^A is employed. The other side of the coin is that our solution requires a two-step encoding. We need to introduce the concept of *trace* of a query $q_{c,p}$ over s, denoted by $\mathrm{Trace}_s(q_{c,p})$. Given the access pattern $\mathrm{Pat}_s(q_{c,p}) = \{m_1 < m_2 < \cdots < m_t\}$ (see Section 2), the trace is defined as the string $\mathrm{Trace}_s(q_{c,p}) = s[m_1] \cdot s[m2] \cdot \cdots \cdot s[m_t]$. We also extend the concept to sets of queries, so that for $\widehat{Q} \subseteq Q$, we have $\mathrm{Pat}_s(\widehat{Q}) = \bigcup_{q \in \widehat{Q}} \mathrm{Pat}_s(q)$, and $\mathrm{Trace}_s(\widehat{Q})$ is defined using the sorted positions in $\mathrm{Pat}_s(\widehat{Q})$.

Then, we define a *canonical ordering* between query sets. We define the predicate $q_{c,p} \prec q_{d,g}$ iff $p < g$ or $p = g \wedge c < d$ over queries, so that we can sort queries inside a single query set. Let $Q_1 = \{q_1 \prec q_2 \prec \cdots \prec q_x\}$ and let $Q_2 = \{q'_1 \prec q'_2 \prec \cdots \prec q'_y\}$ be two distinct query sets. We say that $Q_1 \prec Q_2$ iff either $q_1 \prec q'_1$ or recursively $(Q_1 \setminus \{q_1\}) \prec (Q_2 \setminus \{q'_1\})$.

Given Q^*, its archetype Q^A obeys to the following conditions for the given s:

- it is conflict free and has the same number of queries of Q^*;
- it contains exactly the same stumbling queries of Q^*, and all remaining queries are z-unique (note that they may differ from those in Q^*);
- if $p_1, p_2, \ldots, p_x$ are the positional arguments of queries in Q^*, then the same positions are found in Q^A (while character $c_1, c_2, \ldots, c_x$ may change);
- $\mathrm{Pat}_s(Q^*) = \mathrm{Pat}_s(Q^A)$;
- among those query sets complying with the above properties, it is the minimal w.r.t. to the canonical ordering $\prec$.

Note that Q^* complies with all the conditions above but the last. Therefore, the archetype of Q^* always exists, being either a smaller query set (w.r.t. to $\prec$) or Q^* itself. The encoder can compute Q^A by exhaustive search, since its time complexity is not relevant to the lower bound.

First step: encoding for trace and stumbling queries. As noted above the stumbling queries for Q^* and Q^A are the same, and there are δ of them. Here, we encode the trace together with the set of stumbling queries. The rationale is that the decoder must be able to rebuild the original trace only, whilst encoding of the positions which are not probed is left to the next step, together with z-unique queries. Here is the list of objects to be encoded in order:

(a) The set of stumbling queries expressed as a subset of Q.
(b) The access pattern $\mathrm{Pat}_s(Q^A)$ encoded as a subset of $[n]$, the positions of s.
(c) The *reduced* trace, obtained from $\mathrm{Trace}_s(Q^A)$ by removing all the characters in positions that are targets of stumbling queries. Encoding is performed naively by storing each character using $\log \sigma$ bits. The positions thus removed, relatively to the trace, are stored as a subset of $[|\mathrm{Trace}_s(Q^A)|]$.
(d) For each stumbling query $q_{c,p}$, in the canonical order, an encoded integer i of $\log t$ bits indicating that the ith probe accesses the target of the query.

The decoder starts with an empty string, it reads the access pattern in (b), the set of removed positions in (c), and distributes the contents of the reduced trace (c) into the remaining positions. In order the fill the gaps in (c), it recovers the stumbling queries in (a) and runs each of them, in canonical ordering. Using the information in (d), as proved by Lemma 2, it can discover the target in which to place its symbol c. Since Q^A is conflict free, we are guaranteed that each query will always find a symbol in the probed positions.

Lemma 4. *Let ℓ be the length of $\mathrm{Trace}_s(Q^A)$. The first step encodes information (a)–(d) using at most $\ell \log \sigma + O(n) + \delta \log(|Q|/\delta) - \delta \log(\sigma/t)$ bits.*

Proof. Space occupancy for all objects: (a) uses $\log \binom{|Q|}{\delta} = \delta \log(|Q|/\delta) + O(\delta)$; (b) uses $\log \binom{n}{\ell} \leq n$ bits; (c) uses $(\ell - \delta) \log \sigma$ bits for the reduced trace plus at most ℓ bits for the removed positions; (d) uses $\delta \log t$ bits. □

Second step: encoding of z-unique queries and unprobed positions. We now proceed to the second step, where targets for z-unique queries are encoded along with the unprobed positions. They can be rebuilt using queries in Q^A. To this end, we assume that encoding of Lemma 4 has already been performed and, during decoding, we assume that the trace has been already rebuilt. Recall that γ is the number of z-unique queries. Here is the list of objects to be encoded:

(e) The set of queries in Q^A that are z-unique, expressed as a subset of Q^A according to the canonical ordering $\prec$. Also the set of z-unique answers U is encoded as a subset of $[n]$.

(f) For each z-unique query $q_{c,p}$, in canonical order, the encoded integer p. This gives a multiset of γ integers in $[n]$.

(g) The *reduced* unprobed region of the string, obtained by removing all the characters in positions that are targets of z-unique queries. Encoding is performed naively by storing each character using $\log \sigma$ bits. The positions thus removed, relatively to the unprobed region, are stored as a subset of $[n - \ell]$.

(h) For each z-unique query $q_{c,p}$, in the canonical order, an encoded integer i of $\log z + O(1)$ bits indicating which position in $[p, p + z]$ contains c.

The decoder first obtains Q^A by exhaustive search. It initializes a set of $|Q^A|$ empty couples (c, p) representing the arguments of each query in canonical order. It reads (e) and reuses (a) to obtain the parameters of the stumbling queries inside Q^A. It then reads (f) and fills all the positional arguments of the queries. Then, it starts enumerating all query sets in canonical order that are compatible with the arguments known so far. That is, it generates characters for the arguments of z-unique queries, since the rest is known. Each query set is then tested in the following way. The decoder executes each query by means of the trace. If the execution tries a probe outside the access pattern, the decoder skips to the next query set. If the query conflicts with any other query inside the same query set, or its answer denotes that the query is not z-unique (see Section 2 and (e)), it skips to the next query set. In this way, all the requirements for the archetype are met, hence the first query set that is not skipped is Q^A.

Using Q^A the decoder rebuilds the characters in the missing positions of the reduced unprobed region: it starts by reading positions in (g) and using them to distribute the characters in the reduced region encoded by (g) again. For each z-unique query $q_{c,p} \in Q^A$, in canonical order, the decoder reads the corresponding integer i inside (h) and infers that $s[i + p] = c$. Again, conflict freedom ensures that all queries can be executed and the process can terminate successfully.

Lemma 5. *The second step encodes information* (e)–(h) *using at most* $(n - \ell) \log \sigma + O(n) - \gamma \log(\sigma/z)$ *bits.*

Proof. Space occupancy: (e) uses $\log \binom{|Q^A|}{\gamma} \leq |Q^A|$ bits for the subset plus, recalling from Section 2, $O(n)$ bits for U ; (f) uses $\log \binom{n+\gamma}{\gamma} \leq 2n$ bits; (g)

requires $(n-\ell-\gamma)\log\sigma$ bits for the reduced unprobed region plus $\log\binom{n-\ell}{\gamma}$ bits for the positions removed; (h) uses $\gamma\log z + O(\gamma)$ bits. □

Proof (of Theorem 1). By combining Lemma 4 and Lemma 5 we obtain that for each $s \in S$, $E(s) \leq n\log\sigma + O(n) + \delta\log\left(\frac{t|Q|}{\delta\sigma}\right) - \gamma\log\left(\frac{\sigma}{z}\right)$. We know that $r + \max_{s\in S} E(s) \geq H(S) \geq n\log\sigma - \Theta(n)$, hence the bound follows. □

4 Upper Bounds

Our approach follows substantially the one in [1], but uses two new ingredients, that of *monotone* hashing [2] and *succinct SB-trees* [12], to achieve an improved (and in many cases optimal) result. We first consider these problems in a slightly different framework and give some preliminaries.

Preliminaries. We are given a subset $T \subseteq [\sigma]$, where $|T| = m$. Let $R(i) = |\{j \in T | j < i\}|$ for any $i \in [\sigma]$, and $S(i)$ be the $i+1$st element of T, for $i \in [m]$.

The value of $S(R(p))$ for any p is named the *predecessor* of p inside T. For any subset $T \subseteq [\sigma]$, given access to $S(\cdot)$, a *succinct SB-tree* [12] is a systematic data structure that supports predecessor queries on T, using $O(|T|\log\log\sigma)$ extra bits. For any $c > 0$ such that $|T| = O(\log^c\sigma)$, the succinct SB-tree supports predecessor queries in $O(c)$ time plus $O(c)$ calls to $S(\cdot)$. The data structure relies on a precomputed table of $n^{1-\Omega(1)}$ bits depending only on σ,d not on T.

A *monotone minimal perfect* hash function for T is a function h_T such that $h_T(x) = R(x)$ for all $x \in T$, but $h_T(x)$ can be arbitrary if $x \notin T$. We need a monotone minimal perfect hash function for T, as introduced in [2], that occupies $O(m\log\log\sigma)$ bits and can be evaluated in $O(1)$ time.

Although function $R(\cdot)$ has been studied extensively in the case that T is given explicitly, we consider the situation where T can only be accessed through (expensive) calls to $S(\cdot)$. We also wish to minimize the space used (so e.g. creating an explicit copy of T in a preprocessing stage, and then applying existing solutions, is ruled out). We give the following extension of known results:

Lemma 6. *Let $T \subseteq [\sigma]$ and $|T| = m$. Then, for any $1 \leq k \leq \log\log\sigma$, there is a data structure that supports $R(\cdot)$ in $O(\log\log\sigma)$ time plus $O(1+\log k)$ calls to $S(\cdot)$, and uses $O((m/k)\log\log\sigma)$ bits of space. The data structure uses a pre-computed table (independent of T) of size $\sigma^{1-\Omega(1)}$ bits.*

Proof. We construct the data structure as follows. We store every $(\log\sigma)$th element of T in a y-fast trie [18]. This divides T into *buckets* of $\log\sigma$ consecutive elements. For any bucket B, we store every kth element of T in a succinct SB-tree. The space usage of the y-fast trie is $O(m)$ bits, and that of the succinct SB-tree is $O((m/k)\log\log\sigma)$ bits.

To support $R(\cdot)$, we first perform a query on the y-fast trie, which takes $O(\log\log\sigma)$ time. We then perform a query in the appropriate bucket, which takes $O(1)$ time by looking up a pre-computed table (which is independent of T) of size $\sigma^{1-\Omega(1)}$. The query in the bucket also requires $O(1)$ calls to $S(\cdot)$. We

have so far computed the answer within k keys in T: to complete the query for $R(\cdot)$ we perform binary search on these k keys using $O(\log k)$ calls to $S(\cdot)$. □

Supporting `rank` and `select`. In what follows, we use Lemma 6 choosing $k = 1$ and $k = \log\log\sigma$. We now show the following result, contributing to eq. (1). Note that the first option in Theorem 4 has optimal index size for t probes, for $t \leq \log\sigma/\log\log\sigma$. The second option has optimal index size for t probes, for $t \leq \log\sigma/\log^{(3)}\sigma$, but only for `select`.

Theorem 4. *For any $1 \leq t \leq \sigma$, there exist data structures with complexities:*

(a) `select` *in $O(t)$ probes and $O(t)$ time, and* `rank` *in $O(t)$ probes and $O(t + \log\log\sigma)$ time using a succinct index with $r = O(n(\log\log\sigma + (\log\sigma)/t))$ bits of redundancy. If $\sigma = (\log n)^{O(1)}$, the* `rank` *operation requires only $O(t)$ time.*

(b) `select` *in $O(t)$ probes and $O(t\log\log\sigma)$ time, and* `rank` *in $O(t\log^{(3)}\sigma)$ probes and $O(t\log\log\sigma\log^{(3)}\sigma)$ time, using $r = O(n(\log^{(3)}\sigma + (\log\sigma)/t))$ bits of redundancy for the succinct index.*

Proof. We divide the given string s into contiguous blocks of size σ (assume for simplicity that σ divides $n = |s|$). As in [1,10], we use $O(n)$ bits of space, and incur an additive $O(1)$-time slowdown, to reduce the problem of supporting `rank` and `select` on s to the problem of supporting these operations on a given block B. We denote the individual characters of B by $B[0], \ldots, B[\sigma-1]$.

Our next step is also as in [1]: letting n_c denote the multiplicity of character c in B, we store the bitstring $Z = 1^{n_0}01^{n_1}0\ldots1^{n_{\sigma-1}}0$s of length 2σ, and augment it with the binary `rank` and `select` operations, using $O(\sigma)$ bits in all. Let $c = B[i]$ for some $0 \leq i \leq \sigma-1$, and let $\pi[i]$ be the position of c in a stably sorted ordering of the characters of B (π is a permutation). As in [1], `select`$(c,\cdot)$ is reduced, via Z, to determining $\pi^{-1}(j)$ for some j. As shown in [14], for any $t \in [\sigma]$ permutation π can be augmented with $O(\sigma + \frac{\sigma\log\sigma}{t})$ bits so that $\pi^{-1}(j)$ can be computed in $O(t)$ time plus t evaluations of $\pi(\cdot)$ for various arguments.

If T_c denotes the set of indexes in B containing the character c, we store a minimal monotone hash function h_{T_c} on T_c, for all $c \in [\sigma]$. To compute $\pi(i)$, we probe s to find $c = B[i]$, and observe that $\pi(i) = R(i) + \sum_{i=0}^{c-1} n_i$. The latter term is obtained in $O(1)$ time by operations on Z, and the former term by evaluating $h_{T_c}(i)$. By the result in [2], the complexity of `select`(c,i) is as claimed.

As noted above, supporting `rank`(c,i) on s reduces to supporting `rank` on an individual block B. If T_c is as above, we apply Lemma 6 to each T_c, once with $k = 1$ and once with $k = \log\log\sigma$. Lemma 6 requires some calls to $S(\cdot)$, but this is just `select`$(c,\cdot)$ restricted to B, and is solved as described above. If $\sigma = (\log n)^{O(1)}$, then $|T_c| = (\log n)^{O(1)}$, and we store T_c itself in the succinct SB-tree, which allows us to compute $R(\cdot)$ in $O(1)$ time using a (global, shared) lookup table of size $n^{1-\Omega(1)}$ bits. □

The enhancements described here also lead to more efficient non-systematic data structures. Namely, for $\sigma = \Theta(n^{\varepsilon})$, $0 < \varepsilon < 1$, we match the lower bound of [9, Theorem 4.3]. Moreover, we improve asymptotically both in terms of space and time over the results of [1], employing techniques such as [5,11,17]:

Corollary 2. *There exists a data structure that represents any string s of length n using $nH_k(s) + O(\frac{n \log \sigma}{\log \log \sigma})$ bits, for any $k = o(\frac{\log_\sigma n}{\log \log \sigma})$, supporting* `access` *in $O(1)$ time,* `rank` *and* `select` *in $O(\log \log \sigma)$ time.*

Acknowledgments. We are grateful to a referee for the numerous comments. This work was supported in part by MIUR of Italy under project AlgoDEEP prot. 2008TFBWL4.

References

1. Barbay, J., He, M., Munro, J.I., Rao, S.S.: Succinct indexes for strings, binary relations and multi-labeled trees. In: Proc. SODA 2007, pp. 680–689 (2007); Also, full version available in Internet
2. Belazzougui, D., Boldi, P., Pagh, R., Vigna, S.: Monotone minimal perfect hashing: searching a sorted table with $O(1)$ accesses. In: Proc. SODA 2009, pp. 785–794 (2009)
3. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Series in Telecommunications and Signal Processing). Wiley-Interscience, Hoboken (2006)
4. Demaine, E.D., López-Ortiz, A.: A linear lower bound on index size for text retrieval. J. Algorithms 48(1), 2–15 (2003)
5. Ferragina, P., Venturini, R.: A simple storage scheme for strings achieving entropy bounds. Theor. Comput. Sci. 372(1), 115–121 (2007)
6. Gál, A., Bro Miltersen, P.: The cell probe complexity of succinct data structures. Theor. Comput. Sci. 379, 405–417 (2007)
7. Golynski, A.: Optimal lower bounds for rank and select indexes. TCS 387, 348–359 (2007)
8. Golynski, A.: Upper and Lower Bounds for Text Indexing Data Structures. PhD Thesis, U. Waterloo (2007)
9. Golynski, A.: Cell probe lower bounds for succinct data structures. In: Proc. SODA 2009, pp. 625–632 (2009)
10. Golynski, A., Munro, J.I., Rao, S.S.: Rank/select operations on large alphabets: a tool for text indexing. In: Proc. SODA 2006, pp. 368–373 (2006)
11. González, R., Navarro, G.: Statistical encoding of succinct data structures. In: Lewenstein, M., Valiente, G. (eds.) CPM 2006. LNCS, vol. 4009, pp. 294–305. Springer, Heidelberg (2006)
12. Grossi, R., Orlandi, A., Raman, R., Rao, S.S.: More haste, less waste: Lowering the redundancy in fully indexable dictionaries. In: Proc. STACS 2009, pp. 517–528 (2009)
13. Miltersen, P.B.: Lower bounds on the size of selection and rank indexes. In: Proc. SODA 2005, pp. 11–12 (2005)
14. Munro, J.I., Raman, R., Raman, V., Rao, S.S.: Succinct representations of permutations. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 345–356. Springer, Heidelberg (2003)
15. Pătraşcu, M.: Succincter. In: Proc. FOCS 2008, pp. 305–313 (2008)
16. Raman, R., Raman, V., Rao, S.S.: Succinct indexable dictionaries, with applications to representing k-ary trees, prefix sums and multisets. ACM Transactions on Algorithms 4 (2007)
17. Sadakane, K., Grossi, R.: Squeezing succinct data structures into entropy bounds. In: Proc. SODA 2006, pp. 1230–1239 (2006)
18. Willard, D.E.: Log-logarithmic worst-case range queries are possible in space Theta(N). IPL 17(2), 81–84 (1983)

Approximation Algorithms for Optimal Decision Trees and Adaptive TSP Problems

Anupam Gupta[1,⋆], Viswanath Nagarajan[2], and R. Ravi[3,⋆⋆]

[1] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
[3] Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract. We consider the problem of constructing optimal decision trees: given a collection of tests which can disambiguate between a set of m possible diseases, each test having a cost, and the a-priori likelihood of the patient having any particular disease, what is a good adaptive strategy to perform these tests to minimize the expected cost to identify the disease? We settle the approximability of this problem by giving a tight $O(\log m)$-approximation algorithm.

We also consider a more substantial generalization, the *Adaptive TSP* problem, which can be used to model switching costs between tests in the optimal decision tree problem. Given an underlying metric space, a random subset S of cities is drawn from a known distribution, but S is initially unknown to us—we get information about whether any city is in S only when we visit the city in question. What is a good adaptive way of visiting all the cities in the random subset S while minimizing the expected distance traveled? For this adaptive TSP problem, we give the first poly-logarithmic approximation, and show that this algorithm is best possible unless we can improve the approximation guarantees for the well-known group Steiner tree problem.

1 Introduction

Consider the following two adaptive covering optimization problems:

- *Adaptive TSP under stochastic demands.* (AdapTSP) A traveling salesperson is given a metric space (V, d), distinct subsets $S_1, S_2, \ldots, S_m \subseteq V$ such that S_i appears with probability p_i (and $\sum_i p_i = 1$), and she needs to serve requests at this subset of locations. However, she does not know the identity of the random subset: she can only visit locations, at which time she finds out whether or not that location is part of the subset. What adaptive strategy should she use to minimize the expected time to serve all requests?
- *Optimal Decision Trees.* Given a set of m diseases, there are n binary tests that can be used to disambiguate between these diseases. If the cost of

⋆ Supported in part by NSF awards CCF-0448095 and CCF-0729022, and an Alfred P. Sloan Fellowship.
⋆⋆ Supported in part by NSF grant CCF-0728841.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 690–701, 2010.

performing test $t \in [n]$ is c_t, and we are given the likelihoods $\{p_j\}_{j\in[m]}$ that a typical patient has the disease j, what (adaptive) strategy should the doctor use for the tests to minimize the expected cost to identify the disease?

It can be shown that the optimal decision tree problem is a special case of the former problem (a formal reduction is given in Section 4.) In both these problems we want to devise adaptive strategies, which take into account the revealed information in the queries so far (e.g., cities already visited, or tests already done) to determine the future course of action—i.e., our output must be a decision tree. In contrast, a non-adaptive strategy corresponds to a decision list. While some problems have the property that the best adaptive and non-adaptive strategies have similar performances, both the above problems have a large *adaptivity gap* [9] (the worst ratio between the performance of the optimal non-adaptive and optimal adaptive strategies). Hence it is essential that we seek good *adaptive* strategies.

The *optimal decision tree problem* has long been studied (its NP-hardness was shown in 1976 [22], and many references and applications can be found in [29]). On the algorithms side, $O(\log m)$-approximation algorithms have been given for the special cases where the likelihoods $\{p_j\}$ are all equal, or where the test costs $\{c_t\}$ are all equal [25,8,1,4,29,18]. But the problem has remained open for the case when both the probabilities and the costs are arbitrary—the previous results only imply $O\left(\log \frac{1}{p_{\min}}\right)$ and $O\left(\log(m\,\frac{c_{max}}{c_{min}})\right)$ approximation algorithms for the general case. An $O(\log m)$-approximation for general costs and probabilities would be the best possible unless $NP \subseteq DTIME[n^{O(\log\log n)}]$ [4]; and while the existence of such an algorithm has been posed as an open question, it has not been answered prior to this work.

Apart from being a natural adaptive optimization problem, AdapTSP can be viewed as a generalization of the optimal decision tree problem when there are arbitrary (metric) switching costs between tests. Similar extensions from uniform to metric switching-costs, have been studied for the *multi-armed bandit* problem [15,16]. To the best of our knowledge, the adaptive TSP problem has not been studied previously.

In this paper, we settle the approximability of the optimal decision tree problem:

Theorem 1. *There is an $O(\log m)$-approximation for the optimal decision tree problem with arbitrary test costs and arbitrary probabilities; the problem admits the same approximation ratio even when the tests have non-binary outcomes.*

In fact, this result arises as a special case of the following theorem:

Theorem 2. *There is an $O(\log^2 n \log m)$-factor approximation algorithm for the adaptive TSP problem.*

We also show that AdapTSP is at least as hard to approximate as the well-known group Steiner tree problem [12,21]. Thus AdapTSP is $\Omega(\log^{2-\epsilon} n)$ hard to approximate even on tree metrics; our results are essentially best possible on such metrics, and we lose an extra logarithmic factor to go to general metrics, as in the group Steiner tree problem.

A Word about our Techniques. To solve the AdapTSP problem, we first solve the "Isolation" problem, which seeks to identify which of the m scenarios has materialized; once we know the scenario we can visit its vertices using, say, Christofides' heuristic. The idea behind our algorithm for Isolation is this—suppose each vertex lies in at most half the scenarios; if we visit one vertex in each of the m scenarios using a short tour (which is just group Steiner tree [12]), we'd notice at least one of these vertices to have a demand—this would reduce the number of possible scenarios by 50%. This is necessarily an over-simplified view, and there are many details to handle: we need not visit all scenarios—visiting all but one allows us to infer the last one by exclusion; the expectation in the objective function means we need to solve a *min-latency* version of group Steiner tree; not all vertices need lie in less than half the scenarios. Another major issue is that moreover we do not want our performance to depend on the size of the probabilities, in case some of them are exponentially small, so we cannot just hope to reduce the measure of the remaining scenarios by 50%. Finally, we need to charge our cost directly against the optimal decision tree. All these issues can be resolved to give an $O(\log^2 n \cdot \log m)$ approximation for Isolation (and hence for AdapTSP) with n nodes and m scenarios. The Isolation algorithm also involves an interesting combination of ideas from the group Steiner [12,5] and minimum latency [2,6,10] problems—it uses a greedy strategy that is greedy with respect to two different criteria, namely both the probability measure and the number of scenarios. This idea is formalized in our definition of the *partial latency* group Steiner (LPGST) problem.

For the special case of the optimal decision tree problem, we show that we can use the min-sum set cover problem [11] instead of the min-latency version of group Steiner tree; this avoids an $O(\log^2 n)$ loss in the approximation guarantee, and hence gives us an optimal $O(\log m)$-approximation for the optimal decision tree problem. Our result further reinforces the close connection between the min-sum set cover problem and the optimal decision tree problem that was first noticed by Chakravarthy et al. [4].

Paper Outline. The results on AdapTSP and Isolation appear in Section 3, and the improved algorithm for optimal decision tree appears in Section 4. Due to lack of space, we refer the reader to the full version [20] for all missing proofs. In [20] we also give the hardness result for AdapTSP, and extend our algorithm to obtain a poly-logarithmic approximation algorithm for the related *adaptive traveling repairman problem* (where the objective is expected response time).

1.1 Other Related Work

The optimal decision tree problem has been studied earlier by many authors, with algorithms and hardness results being shown by [22,25,1,8,4,3,18]; as mentioned above, the algorithms in these papers give $O(\log m)$-approximations only when either the probabilities or costs (or both) are polynomially-bounded. Table 1 in Guillory and Bilmes [18] gives a good summary of known approximation ratios; in general, the best approximation guarantees are $O\left(\log \frac{1}{p_{\min}}\right)$ and

$O\left(\log(m\frac{c_{max}}{c_{min}})\right)$. The $O(\log m)$-approximation for arbitrary costs and probabilities solves an open problem from these papers.

There are many results on adaptive optimization. E.g., [13] considered an adaptive set-cover problem; they give an $O(\log n)$-approximation when sets may be chosen multiple times, and an $O(n)$-approximation when each set may be chosen at most once. This was improved in [27] to $O(\log^2 n \log m)$, and subsequently to the best-possible $O(\log n)$-approximation by [26], also using a greedy algorithm. In very recent work [14] generalize adaptive set-cover to so-called 'adaptive submodularity', and give some applications. [7] considered adaptive strategies for stochastic routing problems using a greedy-like algorithm where they solved an LP at each step. [17] studied adaptive transmission in noisy channels (that have multiple states) to maximize the utility (i.e., the difference between success-probability and probing-cost). The adaptivity-gap is large in all these problems, as is the case for the problems considered in this paper, and hence the solutions need to be inherently adaptive.

The AdapTSP problem is related to universal TSP [24,19] and *a priori* TSP [23,30,31] only in spirit—in both the universal and *a priori* TSP problems, we seek a master tour which we shortcut once the demand set is known, and the goal is to minimize the worst-case or expected length of the shortcut tour. The crucial difference is that the demand subset is revealed *in toto* in these two problems, leaving no possibility of adaptivity—this is in contrast to the slow revelation of the demand subset that occurs in AdapTSP.

2 Notation

Throughout this paper, we deal with demand distributions over vertex-subsets that are specified explicitly. I.e. demand distribution $\mathcal{D}$ is specified by m distinct subsets $\{S_i \subseteq V\}_{i=1}^m$ having associated probabilities $\{p_i\}_{i=1}^m$ such that $\sum_{i=1}^m p_i = 1$. This means that the realized subset $D \subseteq V$ of demand-vertices will always be one of $\{S_i\}_{i=1}^m$, where $D = S_i$ with probability p_i (for all $i \in [m]$). We also refer to the subsets $\{S_i\}_{i=1}^m$ as *scenarios*. The following definition captures all adaptive strategies.

Definition 1 (Decision Tree). *A decision tree T in metric (V, d) is a rooted binary tree where each non-leaf node of T is labeled with a vertex $u \in V$, and its two children u_{yes} and u_{no} correspond to the subtrees taken if there **is** demand at u or if there is **no** demand at u. Thus given any realized demand $D \subseteq V$, a unique path P_D is followed in T from the root down to a leaf.*

For metric (V, d) and root $r \in V$, an *r-tour* is a tour that begins and ends at r.

3 The Adaptive TSP and Isolation Problems

The input to *adaptive TSP* is a metric (V, d) with root $r \in V$ and a demand distribution $\mathcal{D}$ over subsets of vertices. The crucial aspect in this model is that information on whether or not there is demand at a vertex v is obtained only

when that vertex v is visited. The objective is to find an adaptive strategy that minimizes the expected time to visit all vertices of the realized scenario drawn from $\mathcal{D}$.

We assume that the distribution $\mathcal{D}$ is specified *explicitly* with a support-size of m. The more general setting would be to consider black-box access to distribution $\mathcal{D}$: however, as shown in [28], AdapTSP under general black-box distributions admits no $o(n)$ approximation if the algorithm is restricted to polynomially many samples. This is the reason we consider an explicit demands model for $\mathcal{D}$. Moreover, AdapTSP under explicit demands still contains interesting special cases such as optimal decision tree problem. One could also consider AdapTSP under independent demand distributions; however in this case there is a trivial solution: visit all vertices having non-zero probability along an approximately minimum TSP tour. In terms of Definition 1, we have:

Definition 2 (The AdapTSP Problem). *Given metric (V, d), root r and demand distribution $\mathcal{D}$, the goal in* AdapTSP *is to compute a decision tree T in metric (V, d) with the added conditions that:*

- *the root of T is labeled with the root vertex r, and*
- *for each scenario $i \in [m]$, the path P_{S_i} followed on input $S_i \in support(\mathcal{D})$ contains all vertices in S_i.*

The objective function is to minimize the expected tour length $\sum_{i=1}^{m} p_i \cdot d(P_{S_i})$, where $d(P_{S_i})$ is the length of the tour that starts at r, visits the vertices on path P_{S_i} in that order, and returns to r.

A closely related problem is the *Isolation* problem. This has the same input as the AdapTSP problem, but the goal is not necessarily to visit all the vertices in the realized scenario, but just to identify the unique scenario that has materialized.

Definition 3 (The Isolation Problem). *Given metric (V, d), root r and demand distribution $\mathcal{D}$, the goal in* Isolation *is to compute a decision tree T in metric (V, d) with the added conditions that:*

- *the root of T is labeled with the root vertex r, and*
- *for each scenario $i \in [m]$, the path P_{S_i} followed on input $S_i \in support(\mathcal{D})$ ends at a distinct leaf-node of T.*

The objective function is to minimize the expected tour length $\mathsf{IsoTime}(T) := \sum_{i=1}^{m} p_i \cdot d(P_{S_i})$, where $d(P_{S_i})$ is the length of the tour that starts at r, visits the vertices on path P_{S_i} in that order, and returns to r.

Note the only difference between this definition and the one for AdapTSP is that the tree path P_{S_i} need not contain all vertices of S_i, and the paths for different scenarios should end at distinct leaf-nodes. The following simple lemma relates these two objectives.

Lemma 1. *An α-approximation algorithm for* Isolation *implies an $\left(\alpha + \frac{3}{2}\right)$ approximation for* AdapTSP.

In the next few subsections, we give an $O(\log^2 n \log m)$-approximation algorithm for Isolation, which by this lemma implies the same result for AdapTSP.

3.1 Approximation Algorithm for the Isolation Problem

Recall that an instance of Isolation is specified by a metric (V, d), a root vertex $r \in V$, and m scenarios $\{S_i\}_{i=1}^m$ with associated probability values $\{p_i\}_{i=1}^m$. At a high level, our approach is a simple divide-and-conquer based one. The algorithm for Isolation is recursive: given an instance, it first develops a strategy to generate several sub-instances from this instance, each given by some proper subset $M \subseteq [m]$ of scenarios, with associated probabilities $\{q_i\}_{i \in M}$ where $\sum_{i \in M} q_i = 1$. (We refer to such a sub-instance of the original Isolation instance as $\langle M, \{q_i\}_{i \in M} \rangle$.) We then recursively build an isolation strategy within each sub-instance. The real question is: *How do we generate these sub-instances so that we can charge these to the cost of the best decision tree?*

A useful subroutine to address this and solve Isolation will be the *partial latency group Steiner* (LPGST) problem, where we are given a metric (V, d), g groups of vertices $\{X_i \subseteq V\}_{i=1}^g$ with associated weights $\{w_i\}_{i=1}^g$, root $r \in V$, and a target $h \leq g$. A group $i \in [g]$ is said to be *covered* (or visited) by r-tour τ if any vertex in X_i is visited, and the *arrival time* of group i is the length of the shortest prefix of τ that contains an X_i-vertex. The arrival times of all uncovered groups are set to be the tour-length. The weighted sum of arrival times of all groups is termed *latency* of the tour, i.e.,

$$\text{latency}(\tau) = \textstyle\sum_{i \text{ covered}} w_i \cdot \text{arrival time}_\tau(X_i) + \sum_{i \text{ uncovered}} w_i \cdot \text{length}(\tau). \quad (3.1)$$

The goal in the LPGST problem is to compute a *minimum latency* tour that *covers at least* h groups. Note that this objective generalizes the standard min-latency objective, where we have to cover *all* groups. In the full version [20] we prove the following result:

Theorem 3. *There is an* $(O(\log^2 n), 4)$*-bicriteria approximation algorithm for* LPGST*. I.e., the tour output by the algorithm visits at least* $\frac{h}{4}$ *groups and has latency at most* $O(\log^2 n)$ *times the optimal latency of a tour that visits* h *groups.*

In this subsection, we present the algorithm for Isolation assuming this result for LPGST. Let us begin by showing the partitioning algorithm.

The Partitioning Algorithm. The algorithm Partition (given below as Algorithm 1) finds an r-tour τ in the metric such that after observing the demands on τ, the *number of scenarios* that are consistent with these observations is only a constant fraction of the total. E.g., if there was a vertex v such that $\approx 50\%$ of the scenarios contained it, then visiting vertex v would reduce the candidate scenarios by $\approx 50\%$, irrespective of the observation at v, giving us a tangible notion of progress. However, each vertex may give a very unbalanced partition or such a vertex may be too expensive to visit, so we may have to visit multiple vertices—this is the basic idea in algorithm Partition.

The Isolation Algorithm. To follow a divide-and-conquer strategy, the final algorithm is fairly natural given the partitioning scheme. The algorithm IsoAlg (given as Algorithm 2) proceeds in several phases. In each phase, it maintains

Algorithm 1. Algorithm Partition($\langle M, \{q_i\}_{i \in M} \rangle$)

1: **let** $g = |M|$. For each $v \in V$, define $F_v := \{i \in M \mid v \in S_i\}$, and
$D_v := \begin{cases} F_v & \text{if } |F_v| \leq \frac{g}{2} \\ M \setminus F_v & \text{if } |F_v| > \frac{g}{2} \end{cases}$
2: **for each** $i \in M$, set $X_i \leftarrow \{v \in V \mid i \in D_v\}$. Note that either the presence of a demand at some vertex v reduces the number of still-possible scenarios by half, or the absence of demand does so. To handle this asymmetry, this step takes scenarios $\{S_i\}_{i \in M}$ and "flips" some vertices to get X_i.
3: **run** the LPGST algorithm (Theorem 3) with metric (V, d) with root r, groups $\{X_i\}_{i \in M}$ with weights $\{q_i\}_{i \in M}$, and target $|M| - 1$.
let $\tau := r, v_1, v_2, \cdots, v_{t-1}, r$ be the r-tour returned.
4: **let** $\{P_k\}_{k=1}^{t}$ be the partition of M where
$P_k := \begin{cases} D_{v_k} \setminus \left(\cup_{j<k} D_{v_j}\right) & \text{if } 1 \leq k \leq t-1 \\ M \setminus \left(\cup_{j<t} D_{v_j}\right) & \text{if } k = t \end{cases}$
5: **return** tour τ and the partition $\{P_k\}_{k=1}^{t}$.

a candidate set M of scenarios such that the realized scenario lies in M. Upon observing demands along the tour produced by algorithm Partition (in Step 2), a new set $M' \subseteq M$ containing the realized scenario is identified such that the number of candidate scenarios reduces by a constant factor (i.e. $|M'| \leq \frac{7}{8} \cdot |M|$); then IsoAlg recurses on scenarios M'. After $O(\log m)$ such phases the realized scenario would be correctly identified.

Algorithm 2. Algorithm IsoAlg$\langle M, \{q_i\}_{i \in M} \rangle$

1: If $|M| = 1$, return this unique scenario as realized.
2: **run** Partition$\langle M, \{q_i\}_{i \in M} \rangle$
let $\tau = (r, v_1, v_2, \cdots, v_{t-1}, r)$ be the r-tour and $\{P_k\}_{k=1}^{t}$ be the partition of M returned.
3: **let** $q'_j := \sum_{i \in P_k} q_i$ **for all** $j \in 1 \ldots t$.
4: **traverse** tour τ and return directly to r after visiting the first (if any) vertex v_{k^*} (for $k^* \in [t-1]$) that determines that the realized scenario is in $P_{k^*} \subseteq M$. If there is no such vertex until the end of the tour τ, then set $k^* \leftarrow t$.
5: **run** IsoAlg$\langle P_{k^*}, \{\frac{q_i}{q'_{k^*}}\}_{i \in P_{k^*}} \rangle$ to isolate the realized scenario within the subset P_{k^*}.

Note that the adaptive Algorithm IsoAlg implicitly defines a decision tree too: create a path $(r, v_1, v_2, \cdots, v_{t-1}, v_t = r)$, and hang the subtrees created in the recursive call on each instance $\langle P_k, \{\frac{q_i}{q'_k}\} \rangle$ from the respective node v_k.

Remark: We use the LPGST problem as a subroutine in solving Isolation, instead of the seemingly simpler *density group Steiner* problem [5]. The reason behind this is that we measure progress in the recursive scheme IsoAlg as the *number* of candidate scenarios, whereas in computing the objective we need to use the *probabilities* of scenarios. This is also the reason why our final algorithm interleaves both greedy objectives: number and probabilities of scenarios. It is

not clear whether a greedy algorithm w.r.t. only one of these objectives can achieve the same approximation ratio.

3.2 The Analysis for Algorithm IsoAlg

We now prove the performance guarantee and correctness of Algorithm IsoAlg. By the construction in algorithm IsoAlg, it follows that the realized scenario is identified after $O(\log m)$ recursive calls to IsoAlg. To complete the analysis, we need two additional ideas: (1) relating the LPGST and Isolation objective-values in each call to IsoAlg (which bounds the cost in a single sub-instance), and (2) establishing a subadditivity property of Isolation, that bounds the expected cost across all 'parallel' sub-instances (i.e. those in the same phase).

For any instance $\mathcal{J}$ of the isolation problem, let $\mathsf{IsoTime}^*(\mathcal{J})$ denote its optimal value. Missing proofs in the following analysis can be found in [20].

Claim 1. *For any instance $\mathcal{J} = \langle M, \{q_i\}_{i \in M} \rangle$, the optimal value of the* LPGST *instance considered in Step 3 of Algorithm* Partition$(\mathcal{J})$ *is at most* $\mathsf{IsoTime}^*(\mathcal{J})$*—i.e., the* LPGST *instance costs at most the isolation instance.*

Proof: Let T be an optimal decision tree corresponding to Isolation instance $\mathcal{J}$, and hence $\mathsf{IsoTime}^*(\mathcal{J}) = \mathsf{IsoTime}(T)$. Note that by definition of the sets $\{F_v, D_v\}_{v \in V}$, any internal node in T labeled vertex v has its two children v_{yes} and v_{no} corresponding to the realized scenario being in either F_v or $M \setminus F_v$ (in that order), or equivalently, D_v or $M \setminus D_v$ (now not necessarily in that order).

Consider an r-tour σ that starts at the root r of T and moves from each node v to its unique child that corresponds to $M \setminus D_v$, until it reaches a leaf-node l, when it returns to r. Let the vertices in this tour σ be $r, u_1, u_2, \cdots, u_j, r$ (where u_j is the last vertex visited in T before leaf l). Since T is a feasible decision tree for the isolation instance, the leaf l is labeled by a unique scenario $a \in M$ such that when T is run under demands S_a, it traces path from the root to leaf node l. Moreover, every scenario $b \in M \setminus \{a\}$ gives rise to a root-leaf path that diverges from the root-l path (i.e. σ). From the way we constructed σ, the scenarios that diverged from σ were precisely $\cup_{k=1}^{j} D_{u_k}$, and hence $\cup_{k=1}^{j} D_{u_k} = M \setminus \{a\}$.

Now consider the r-tour σ as a potential solution to the LPGST instance in Step 3. Since $|\cup_{k=1}^{j} D_{u_k}| = |M| - 1$, the definition of the X_i's says that this path hits $|M| - 1$ of these sets X_i, so it is indeed feasible. Also, the definition of the isolation cost implies that the latency of this tour σ is at most the isolation cost $\mathsf{IsoTime}(T) = \mathsf{IsoTime}^*(\mathcal{J})$. ■

If we use a $(\rho_{\mathsf{LPGST}}, 4)$-bicriteria LPGST algorithm, we get the following claim:

Claim 2. *The latency of tour τ returned by Algorithm* Partition *is at most $\rho_{\mathsf{LPGST}} \cdot \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i \in M} \rangle)$. Furthermore, the resulting partition $\{P_k\}_{k=1}^{t}$ has each $|P_k| \leq \frac{7}{8}|M|$ for each $k \in [t]$, when $|M| \geq 2$.*

Proof: By Claim 1, the optimal value of the LPGST instance in Step 3 of algorithm Partition is at most $\mathsf{IsoTime}^*(\langle M, \{q_i\}_{i \in M} \rangle)$; now the approximation guarantee from Theorem 3 implies that the latency of the solution tour τ is at most ρ_{LPGST} times that. This proves the first part of the claim.

Consider $\tau := \langle r = v_0, v_1, \cdots, v_{t-1}, v_t = r \rangle$ the tour returned by the LPGST algorithm in Step (3) of algorithm Partition; and $\{P_k\}_{k=1}^t$ the resulting partition. Claim 1 and Theorem 3 imply that the number of groups covered by τ is $|\cup_{k=1}^{t-1} D_{v_k}| \geq \frac{|M|-1}{4} \geq \frac{|M|}{8}$ (when $|M| \geq 2$). By definition of the sets D_v, it holds that $|D_v| \leq |M|/2$ for all $v \in V$; moreover, since all but the last part P_k is a subset of some D_v, it holds that $|P_k| \leq \frac{|M|}{2}$ for $1 \leq k \leq t-1$. Moreover, the set P_t has size $|P_t| = |M \setminus (\cup_{j<t} D_{v_j})| \leq \frac{7}{8}|M|$. ■

Of course, we don't really care about the latency of the tour *per se*, we care about the cost incurred in isolating the realized scenario. But the two are related (by their very construction), as the following claim formalizes:

Claim 3. *At the end of Step 4 of* $\mathsf{IsoAlg}\langle M, \{q_i\}_{i \in M}\rangle$*, the realized scenario lies in* P_{k^*}*. The expected distance traversed in this step* $\leq 2\rho_{\mathsf{LPGST}} \cdot \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i \in M}\rangle)$.

Now, the following simple claim captures the "sub-additivity" of $\mathsf{IsoTime}^*$.

Claim 4. *For any instance* $\langle M, \{q_i\}_{i \in M}\rangle$ *and any partition* $\{P_k\}_{k=1}^t$ *of* M*,*

$$\sum_{k=1}^t q'_k \cdot \mathsf{IsoTime}^*(\langle P_k, \{\tfrac{q_i}{q'_k}\}_{i \in P_k}\rangle) \leq \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i \in M}\rangle), \tag{3.2}$$

where $q'_k = \sum_{i \in P_k} q_i$ *for all* $1 \leq k \leq t$.

Proof: Let T denote the optimal decision tree for the instance $\mathcal{J}_0 := \langle M, \{q_i\}_{i \in M}\rangle$. For each $k \in [t]$, consider instance $\mathcal{J}_k := \langle P_k, \{\frac{q_i}{q'_k}\}_{i \in P_k}\rangle$; one feasible adaptive strategy for instance $\mathcal{J}_k$ is obtained by taking the decision tree T and considering only paths to the leaf-nodes labeled by $\{i \in P_k\}$. Note that this is a feasible solution since T isolates all scenarios $\cup_{k=1}^t P_k$. Moreover, the expected cost of such a strategy for $\mathcal{J}_k$ is $\sum_{i \in P_k} \frac{q_i}{q'_k} \cdot d(\pi_i)$ where π_i denotes the tour traced by T under scenario $i \in P_k$. Hence $\mathsf{Opt}(\mathcal{J}_k) \leq \sum_{i \in P_k} \frac{q_i}{q'_k} \cdot d(\pi_i)$. Summing over all parts $k \in [t]$, we get

$$\sum_{k=1}^t q'_k \cdot \mathsf{Opt}(\mathcal{J}_k) \leq \sum_{k=1}^t q'_k \cdot \sum_{i \in P_k} \frac{q_i}{q'_k} \cdot d(\pi_i) = \sum_{i \in M} q_i \cdot d(\pi_i) = \mathsf{Opt}(\mathcal{J}_0), \tag{3.3}$$

where the penultimate equality uses the fact that $\{P_k\}_{k=1}^t$ partitions M. ■

Given the above claims, we finally bound the expected cost of the strategy given by our algorithm.

Theorem 4. *The expected length of the strategy given by* $\mathsf{IsoAlg}\langle M, \{q_i\}_{i \in M}\rangle$ *is at most:*

$$2\rho_{\mathsf{LPGST}} \cdot \log_{8/7} |M| \cdot \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i \in M}\rangle).$$

Proof: We prove this by induction on $|M|$. The base case of $|M| = 1$ is trivial, since zero length is traversed, and hence we consider $|M| \geq 2$. Let instance $\mathcal{I}_0 := \langle M, \{q_i\}_{i \in M}\rangle$. For $k \in [t]$, consider the sub-instance $\mathcal{I}_k := \langle P_k, \{\frac{q_i}{q'_k}\}_{i \in P_k}\rangle$, where $q'_k = \sum_{i \in P_k} q_i$. By the inductive hypothesis, for any $k \in [t]$, the expected

length of $\mathsf{IsoAlg}(\mathcal{I}_k)$ is at most $2\rho_{\mathsf{LPGST}} \cdot \log_{8/7}|P_k| \cdot \mathsf{IsoTime}^*(\mathcal{I}_k) \le 2\rho_{\mathsf{LPGST}} \cdot (\log_{8/7}|M| - 1) \cdot \mathsf{IsoTime}^*(\mathcal{I}_k)$, since $|P_k| \le \frac{7}{8}|M|$ (from Claim 2 as $|M| \ge 2$).

By Claim 3, the expected length traversed in Step 4 of $\mathsf{IsoAlg}(\mathcal{I}_0)$ is at most $2\rho_{\mathsf{LPGST}} \cdot \mathsf{IsoTime}^*(\mathcal{I}_0)$. The probability of recursing on $\mathcal{I}_k$ is exactly q'_k for each $k \in [t]$, hence the expected length of $\mathsf{IsoAlg}(\mathcal{I}_0)$ is at most:

$$\begin{aligned}
&2\rho_{\mathsf{LPGST}} \cdot \mathsf{IsoTime}^*(\mathcal{I}_0) + \textstyle\sum_{k=1}^{t} q'_k \cdot (\text{exp. length of } \mathsf{IsoAlg}(\mathcal{I}_k)) \\
&\le 2\rho_{\mathsf{LPGST}} \cdot \mathsf{IsoTime}^*(\mathcal{I}_0) + \textstyle\sum_{k=1}^{t} q'_k \cdot 2\rho_{\mathsf{LPGST}} \cdot (\log_{8/7}|M| - 1) \cdot \mathsf{IsoTime}^*(\mathcal{I}_k) \\
&\le 2\rho_{\mathsf{LPGST}} \cdot \mathsf{IsoTime}^*(\mathcal{I}_0) + 2\rho_{\mathsf{LPGST}} \cdot (\log_{8/7}|M| - 1) \cdot \mathsf{IsoTime}^*(\mathcal{I}_0) \\
&= 2\rho_{\mathsf{LPGST}} \cdot \log_{8/7}|M| \cdot \mathsf{IsoTime}^*(\mathcal{I}_0)
\end{aligned}$$

where the third inequality uses Claim 4. ■

Applying Theorem 4 on the original Isolation instance gives an $O(\rho_{\mathsf{LPGST}} \cdot \log m)$-approximately optimal algorithm; using Theorem 3 implies $\rho_{\mathsf{LPGST}} = O(\log^2 n)$.

Theorem 5. *There is an $O(\log^2 n \cdot \log m)$-approximation algorithm for* Isolation *with n vertices and m scenarios.*

Combining this theorem with Lemma 1 we obtain Theorem 2. A comment about the optimality of these results: we show in [20] that AdapTSP (and by Lemma 1, Isolation as well) is as hard to approximate as the group Steiner problem, and hence any improvement here will make progress on that problem too.

4 Optimal Decision Tree Problem

In the *optimal decision tree problem* [25,1,3], we are given a set of m diseases with associated probabilities $\{p_i\}_{i=1}^m$ that sum to 1, and a collection $\{T_j\}_{j=1}^n$ of n binary tests with non-negative costs $\{c_j\}_{j=1}^n$. Each test $j \in [n]$ is a subset T_j of the diseases that correspond to passing the test; so performing test j distinguishes between diseases T_j and $[m] \setminus T_j$.

Definition 4. *A* test strategy *S is a binary tree where each internal node is labeled by a test, and each leaf node is labeled by a disease such that:*

- *For each disease $i \in [m]$ define a path π_i in S from the root node to some leaf as follows. At any internal node, if i passes the test then π_i follows the right branch; if it fails the test then π_i follows the left branch.*
- *For each $i \in [m]$, the path π_i ends at a leaf labeled disease i.*

The cost *L_i of a disease $i \in [m]$ is the sum of test-costs along path π_i; and the cost of the test strategy S is $\sum_{i=1}^m p_i \cdot L_i$. The objective in the optimal decision tree problem is to compute a test strategy of minimum cost.*

Observe that the optimal decision tree problem is a special case of Isolation: given an instance of optimal decision tree (as above), consider a metric (V, d) induced by a weighted star with center r and n leaves corresponding to the tests. For each $j \in [n]$, we set $d(r, j) = \frac{c_j}{2}$. The demand scenarios are as follows: for each $i \in [m]$, scenario i is $\{j \in [n] \mid i \in T_j\}$. It is easy to see that this Isolation instance corresponds exactly to the optimal decision tree instance. Based on

this, it suffices to focus on Isolation in weighted star-metrics, and we obtain the following improved bound (see [20]).

Theorem 6. *There is an* $\big(O(1), O(1)\big)$ *bicriteria approximation algorithm for* LPGST *on weighted star metrics.*

Setting $\rho_{\mathsf{LPGST}} = O(1)$ (from Theorem 6) in the analysis of Section 3.2, we obtain an $O(\rho_{\mathsf{LPGST}} \cdot \log m) = O(\log m)$ approximation for Isolation on weighted star-metrics. This completes the proof of Theorem 1 for binary outcomes.

Multiway tests. Chakravarthy et al. [3] considered the optimal decision tree problem when the outcomes of tests are multiway (not just binary), and gave an $O(\log m)$-approximation under unit probabilities and costs. We observe that our algorithm can be easily extended to this problem with non-uniform probabilities and costs. In this setting (when each test has at most l outcomes), any test $j \in [n]$ induces a partition $\{T_j^k\}_{k=1}^{l}$ of $[m]$, and performing test j determines which of the parts the realized disease lies in. Firstly note that this problem is also a special case of Isolation. As before consider a metric (V, d) induced by a weighted star with center r and n leaves corresponding to the tests. For each $j \in [n]$, we set $d(r, j) = \frac{c_j}{2}$. Additionally for each $j \in [n]$, introduce l copies of test-vertex j, labeled $(j, 1), \cdots, (j, l)$, at zero distance from each other. The demand scenarios are defined naturally: for each $i \in [m]$, scenario i is $\{(j, k) \mid i \in T_j^k\}$. Clearly this Isolation instance is equivalent to the (multiway) test strategy instance. Since the resulting metric is still a weighted star (we only made vertex copies), Theorem 6 and the algorithm of Section 3.2 imply an $O(\log m)$-approximation for this multiway decision tree problem. Thus we obtain Theorem 1.

References

1. Adler, M., Heeringa, B.: Approximating Optimal Binary Decision Trees. In: Goel, A., Jansen, K., Rolim, J.D.P., Rubinfeld, R. (eds.) APPROX and RANDOM 2008. LNCS, vol. 5171, pp. 1–9. Springer, Heidelberg (2008)
2. Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, W.R., Raghavan, P., Sudan, M.: The minimum latency problem. In: STOC, pp. 163–171 (1994)
3. Chakaravarthy, V., Pandit, V., Roy, S., Sabharwal, Y.: Approximating Decision Trees with Multiway Branches. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 210–221. Springer, Heidelberg (2009)
4. Chakravarthy, V.T., Pandit, V., Roy, S., Awasthi, P., Mohania, M.: Decision Trees for Entity Identification: Approximation Algorithms and Hardness Results. In: PODS (2007)
5. Charikar, M., Chekuri, C., Goel, A., Guha, S.: Rounding via trees: deterministic approximation algorithms for group Steiner trees and k median. In: STOC, pp. 114–123 (1998)
6. Chaudhuri, K., Godfrey, B., Rao, S., Talwar, K.: Paths, trees, and minimum latency tours. In: FOCS, pp. 36–45 (2003)
7. Chawla, S., Roughgarden, T.: Single-source stochastic routing. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX 2006 and RANDOM 2006. LNCS, vol. 4110, pp. 82–94. Springer, Heidelberg (2006)
8. Dasgupta, S.: Analysis of a greedy active learning strategy. In: Advances in Neural Information Processing Systems, NIPS (2004)

9. Dean, B.C., Goemans, M.X., Vondrák, J.: Approximating the stochastic knapsack problem: The benefit of adaptivity. In: FOCS, pp. 208–217 (2004)
10. Fakcharoenphol, J., Harrelson, C., Rao, S.: The k-traveling repairmen problem. ACM Transactions on Algorithms 3(4) (2007)
11. Feige, U., Lovász, L., Tetali, P.: Approximating min sum set cover. Algorithmica 40(4), 219–234 (2004)
12. Garg, N., Konjevod, G., Ravi, R.: A Polylogarithmic Approximation Algorithm for the Group Steiner Tree Problem. Journal of Algorithms 37(1), 66–84 (2000)
13. Goemans, M., Vondrák, J.: Stochastic covering and adaptivity. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, pp. 532–543. Springer, Heidelberg (2006)
14. Golovin, D., Krause, A.: Adaptive Submodularity: A New Approach to Active Learning and Stochastic Optimization (2010), http://arxiv.org/abs/1003.3967
15. Guha, S., Munagala, K.: Approximation algorithms for budgeted learning problems. In: STOC, pp. 104–113. ACM, New York (2007); Full version as Sequential Design of Experiments via Linear Programming, http://arxiv.org/abs/0805.2630v1
16. Guha, S., Munagala, K.: Multi-armed bandits with metric switching costs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5556, pp. 496–507. Springer, Heidelberg (2009)
17. Guha, S., Munagala, K., Sarkar, S.: Information acquisition and exploitation in multichannel wireless networks. CoRR, abs/0804.1724 (2008)
18. Guillory, A., Bilmes, J.: Average-Case Active Learning with Costs. In: Gavaldà, R., Lugosi, G., Zeugmann, T., Zilles, S. (eds.) ALT 2009. LNCS, vol. 5809, pp. 141–155. Springer, Heidelberg (2009)
19. Gupta, A., Hajiaghayi, M.T., Räcke, H.: Oblivious network design. In: SODA, pp. 970–979 (2006)
20. Gupta, A., Nagarajan, V., Ravi, R.: Approximation Algorithms for Optimal Decision Trees and Adaptive TSP Problems (full version). ArXiv (2010), http://arxiv.org/abs/1003.0722
21. Halperin, E., Krauthgamer, R.: Polylogarithmic inapproximability. In: STOC, pp. 585–594 (2003)
22. Hyafil, L., Rivest, R.L.: Constructing optimal binary decision trees is NP-complete. Information Processing Lett. 5(1), 15–17 (1976/1977)
23. Jaillet, P.: A priori solution of a travelling salesman problem in which a random subset of the customers are visited. Operations Research 36(6) (1988)
24. Jia, L., Lin, G., Noubir, G., Rajaraman, R., Sundaram, R.: Universal approximations for TSP, Steiner tree, and set cover. In: STOC, pp. 386–395 (2005)
25. Kosaraju, S.R., Przytycka, T.M., Borgstrom, R.S.: On an Optimal Split Tree Problem. In: Dehne, F., Gupta, A., Sack, J.-R., Tamassia, R. (eds.) WADS 1999. LNCS, vol. 1663, pp. 157–168. Springer, Heidelberg (1999)
26. Liu, Z., Parthasarathy, S., Ranganathan, A., Yang, H.: Near-optimal algorithms for shared filter evaluation in data stream systems. In: SIGMOD, pp. 133–146 (2008)
27. Munagala, K., Srivastava, U., Widom, J.: Optimization of continuous queries with shared expensive filters. In: PODS, pp. 215–224 (2007)
28. Nagarajan, V.: Approximation Algorithms for Sequencing Problems. PhD thesis, Tepper School of Business, Carnegie Mellon University (2009)
29. Nowak, R.D.: The geometry of generalized binary search. arXiv.org:0910.4397 (2009)
30. Schalekamp, F., Shmoys, D.: Algorithms for the universal and a priori TSP. Operations Research Letters 36(1), 1–3 (2008)
31. Shmoys, D., Talwar, K.: A Constant Approximation Algorithm for the a priori Traveling Salesman Problem. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO 2008. LNCS, vol. 5035, pp. 331–343. Springer, Heidelberg (2008)

Concurrent Knowledge Extraction in the Public-Key Model*

Andrew C. Yao[1], Moti Yung[2], and Yunlei Zhao[3,**]

[1] ITCS, Tsinghua University, Beijing, China
[2] Google Inc. and Columbia University, New York, USA
[3] Software School, Fudan University, Shanghai, China
ylzhao@fudan.edu.cn

Abstract. Knowledge extraction is a fundamental notion, modeling machine possession of values (witnesses) in a computational complexity sense and enabling one to argue about the internal state of a party in a protocol without probing its internal secret state. However, when transactions are concurrent (e.g., over the Internet) with players possessing public-keys (as is common in cryptography), assuring that entities "know" what they claim to know, where adversaries may be well coordinated across different transactions, turns out to be much more subtle and in need of re-examination. Here, we investigate how to formally treat knowledge possession by parties (with registered public-keys) interacting over the Internet. Stated more technically, we look into the relative power of the notion of "concurrent knowledge-extraction" (CKE) in the concurrent zero-knowledge (CZK) bare public-key (BPK) model where statements being proven can be dynamically and adaptively chosen by the prover.

We show the potential vulnerability of man-in-the-middle (MIM) attacks turn out to be a real security threat to existing natural protocols running concurrently in the public-key model, which motivates us to introduce and formalize the notion of CKE, alone with clarifications of various subtleties. Then, both generic (based on standard polynomial assumptions), and efficient (employing complexity leveraging in a novel way) implementations for $\mathcal{NP}$ are presented for constant-round (in particular, round-optimal) concurrently knowledge-extractable concurrent zero-knowledge (CZK-CKE) arguments in the BPK model. The efficient implementation can be further practically instantiated for specific number-theoretic language.

1 Introduction

Zero-knowledge (ZK) protocols allow a prover to assure a verifier of validity of theorems without giving away any additional knowledge (i.e., computational advantage) beyond validity. This notion was introduced in [14], and its generality was demonstrated in [13]. Traditional notion of ZK considers the security in a stand-alone (or sequential) execution of the protocol. Motivated by the use of such protocols in an asynchronous

* This work was supported in part by the National Basic Research Program of China Grant Nos.2007CB807900, 2007CB807901, the National Natural Science Foundation of China Grant Nos.60553001, 60703091, and the QiMingXing Program of Shanghai.

** Corresponding author.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 702–714, 2010.

network like the Internet, where many protocols run simultaneously, studying security properties of ZK protocols in such concurrent settings has attracted much research efforts in recent years. Informally, a ZK protocol is called concurrent zero-knowledge if concurrent instances are all (expected) polynomial-time simulatable, namely, when a possibly malicious verifier concurrently interacts with a polynomial number of honest prover instances and schedules message exchanges as it wishes.

The concept of "proof of knowledge" (POK), informally discussed in [14], was then formally treated in [1,11,2]. POK systems, especially zero-knowledge POK (ZKPOK) systems, play a fundamental role in the design of cryptographic schemes, enabling a formal complexity theoretic treatment of what does it mean for a machine to "know" something. Roughly speaking, a "proof of knowledge" means that a possibly malicious prover can convince the verifier that an $\mathcal{NP}$ statement is true if and only if it, in fact, "knows" (i.e., possesses) a witness to the statement (rather than merely conveying the fact that a corresponding witness exists). With the advancement of cryptographic models where parties first publish public-keys (e.g., for improving round complexity [5]) and then may choose the statements to prove, knowledge extraction becomes more subtle (due to possible dependency on published keys), and needs re-examination. Here, we investigate the relative power of the notion of "concurrent knowledge-extraction" in the concurrent zero-knowledge BPK model with adaptive input selection.

The BPK model, introduced in [4], is a natural cryptographic model. A protocol in this model simply assumes that all verifiers have each deposited a public key in a public file (which are referred to as the *key generation stage*), before user interactions take place (which are referred to as the *proof stage*). No assumption is made on whether the public-keys deposited are unique or valid (i.e., public keys can even be "nonsensical," where no corresponding secret-keys exist or are known). In many cryptographic settings, availability of a public key infrastructure (PKI) is assumed or required, and in these settings the BPK model is, both, natural and attractive (note that the BPK model is, in fact, a weaker version of PKI where in the later added key certification is assumed). It was pointed out by Micali and Reyzin [16] that the BPK model is, in fact, applicable to interactive systems in general.

Verifier security in the BPK model (against malicious provers) turned out to be more involved than anticipated, as was demonstrated by Micali and Reyzin [16] who showed that under standard intractability assumptions there are four distinct meaningful notions of soundness, i.e., from weaker to stronger: one-time, sequential, concurrent and resettable soundness. Here, we focus on concurrent soundness, which, roughly speaking, means that a possibly malicious probabilistic polynomial-time (PPT) prover P^* cannot convince the honest verifier V of a *false* statement even when P^* is allowed multiple interleaving interactions with V in the public-key model. They also showed that any black-box ZK protocol with concurrent soundness in the BPK model (for non-trivial languages outside $\mathcal{BPP}$) must run at least four rounds [16].

Concurrent soundness only guarantees that concurrent interactions cannot help a malicious prover validate a *false* statement in the public-key model. However, it does *not* prevent a malicious prover from validating a *true* statement *but* without knowing any witness for the statement being proved. This potential vulnerability is not merely a theoretical concern: In fact, most concurrent ZK protocols in the BPK model involve a

sub-protocol in which the verifier proves to the prover the knowledge of the secret-key corresponding to its public-key. A malicious prover, in turn, can (as we show) exploit these sub-proofs by the verifier in other sessions, without possessing a witness to these sessions' statements. This issue, in turn, motivates the need for careful definitions and for achieving concurrent verifier security for concurrent ZK in the BPK model for adaptively chosen proofs, so that one can remedy the above security vulnerability.

Our contributions. We first investigate the subtleties of concurrent verifier security in the public-key model in the case of proof of knowledge for dynamically chosen input languages. Specifically, we show concurrent interleaving and malleating attacks against some existing natural protocols running concurrently in the BPK model, which shows that concurrent soundness and normal arguments of knowledge (and also traditional concurrent non-malleability) do not guarantee concurrent verifier security in the public-key model.

Then, we formulate concurrent verifier security that remedies the vulnerability as demonstrated by the concrete attacks which are of the concurrent man-in-the-middle (CMIM) nature, along with subtlety clarifications and discussion. The security notion defined is named `concurrent knowledge-extraction (CKE)` in the public-key model, which essentially means that for adaptively chosen statements whose validations are successfully conveyed by a possibly malicious prover to an honest verifier by concurrent interactions, the prover must "know" the corresponding witnesses in a sense that the knowledge known by the prover is "independent" of honest verifier's secret-key.

We then present both generic (based on standard polynomial assumptions) and efficient (employing complexity leveraging in a novel way) black-box implementations of constant-round (in particular, round-optimal) CZK-CKE arguments for $\mathcal{NP}$ in the BPK model. The efficient implementation can be, further, practically instantiated for specific important number-theoretic languages.

2 Preliminaries

In this section, we briefly recall some basic tools and definitions.

Commitments. Commitment schemes enable a party, called the *sender*, to bind itself to a single value in the initial *commitment* stage, while keeping it unknown to the *receiver* (this property is called *hiding*). Furthermore, when the commitment is opened in a later *decommitment* stage, it is guaranteed that the "opening" can yield only the single value determined in the commitment phase (this property is called *binding*).

One-round perfectly-binding commitments can be based on any one-way permutation (OWP) [13], whereas tow-round statistically-binding commitments can be based on any one-way function (OWF) [17]. In addition, practical statistically-binding commitments can be implemented under the decisional Diffie-Hellman (DDH) assumption. On the other hand, one-round statistically-hiding commitments can be based on any collision-resistant hash function [15]. Two-round statistically-hiding commitments can be based on any claw-free collection with efficiently recognizable indices [11], and three-round statistically-hiding commitments can be based on any OWF admitting Σ-protocols [22].

Σ-protocols and Σ_{OR}-protocols. Informally, a Σ-protocol is itself a 3-round public-coin *special* honest verifier zero-knowledge (SHVZK) protocol with special soundness in the knowledge-extraction sense. A Σ-protocol is called computational/statistical Σ-protocol, if it is computational/statistical SHVZK. A very large number of Σ-protocols have been developed in the literature. In particular, (the parallel repetition of) Blum's protocol for DHC [3] is a computational Σ-protocol for $\mathcal{NP}$, and most practical Σ-protocols for number-theoretical languages are of *perfect* SHVZK property. One basic construction with Σ-protocols is the OR of a real and simulated transcript, called Σ_{OR} [6], that is a concrete witness indistinguishability protocol.

Witness Indistinguishability (WI). A protocol is called WI (resp., statistical WI) for an $\mathcal{NP}$-language L, if the views of any PPT malicious verifier V^* in two runs of the protocol, w.r.t. the same common input $x \in L$ and the same auxiliary input $z \in \{0,1\}^*$ to V^* but (possibly) different private witnesses to the prover, are computationally (resp., statistically) indistinguishable. WI is preserved under concurrent composition.

In this work, we employ, in a critical way, constant-round *statistical* WI argument/proof of knowledge (WIA/POK). We briefly note two simple ways to implement statistical WIA/POK. First, for any statistical/perfect Σ-protocol, the OR-proof (i.e., the Σ_{OR}-protocol [6]) is statistical/perfect WIPOK. The second approach is to modify Blum's protocol for DHC [3] (that is computational WIPOK) into constant-round statistical WIAOK, by replacing the statistically-binding commitments used in the first-round of Blum's protocol by constant-round *statistically-hiding* commitments.

Strong WI (SWI) [11]. A protocol $\langle P, V\rangle$ for a language L (with $\mathcal{NP}$-relation R_L) is called SWI, if the views of any PPT malicious verifier V^* in two runs of the protocol, $\langle P(w_0), V^*(z_0)\rangle(x_0)$ and $\langle P(w_1), V^*(z_1)\rangle(x_1)$, are indistinguishable, whenever the distributions (x_0, z_0) and (x_1, z_1) are indistinguishable (where $(x_b, w_b) \in R_L$ for $b \in \{0,1\}$). Any ZK protocol is itself SWI [11]. Different from regular WI, SWI is not preserved under concurrent composition [12]. But, an SWI protocol can be easily transferred into a regular WI protocol: On common input x and private witness w, the prover commits w to c_w, and then proves that the value committed to c_w is a valid witness for $x \in L$. Such a protocol is called *commit-then-SWI*, which is regular WI for L.

The BPK model *with adaptive language selection*. We say a class of languages $\mathcal{L}$ is *admissible* to a protocol $\langle P, V\rangle$ if the protocol can work (i.e., be instantiated) for any language $L \in \mathcal{L}$. Typically, $\mathcal{L}$ could be the set of all $\mathcal{NP}$-languages (via $\mathcal{NP}$-reduction in case $\langle P, V\rangle$ can work for an $\mathcal{NP}$-complete language) or the set of any languages admitting Σ-protocols (in this case $\langle P, V\rangle$ could be instantiated for any language in $\mathcal{L}$ efficiently without going through general $\mathcal{NP}$-reductions). For protocols in the BPK model, let R_{KEY} be an $\mathcal{NP}$-relation validating the public-key and secret-key pair (PK, SK) generated by any honest verifier, i.e., $R_{KEY}(PK, SK) = 1$ indicates that SK is a valid secret-key corresponding to PK.

In this work, for concurrent verifier security of a protocol in the BPK model, we consider an s-concurrent malicious prover P^* that, on a system parameter n, interacts with honest verifier instances in at most $s(n)$ sessions, where $s(\cdot)$ is a polynomial. Furthermore, different from the traditional BPK model formulation [4,16], we assume P^* can set the admissible languages (to be proved to honest verifiers) *that may potentially*

depend on honest verifiers' public-keys. Though it may be more difficult to achieve concurrent verifier security against adversaries with adaptive language selection in the BPK model, this is a far more realistic model for cryptographic protocols running concurrently in the public-key model where mixing the public-key structure as part of the language is a natural adversarial strategy. For any $(PK, SK) \in R_{KEY}$, we denote by $view_{P^*}^{V(SK)}(1^n, z, PK)$ the random variable describing the view of P^* specific to PK, which includes its random tape, the auxiliary string z, the public-key PK, and all messages it receives from the instances of the honest verifier V of secret-key SK.

3 Concurrent Knowledge-Extraction: Motivation, Formulation and Discussion

We show a concurrent interleaving and malleating attack on the concurrent ZK protocol of [7,23] that is both *concurrently sound* and *normal argument of knowledge* (AOK) in the BPK model, in which by concurrent interactions a malicious prover P^* can (with probability 1) convince an honest verifier of a true (*public-key related*) statement but without knowing any witness to the statement being proved. Due to space limitation, the reader is referred to the full paper [20] for the attack details. This shows that concurrent soundness and normal AOK do not guarantee that an adversary does "know" what it concurrently claims to know against an honest verifier in the public-key model. This concrete attack (on *naturally existing* concurrently sound CZKAOK in the BPK model) serves a good motivation for understanding "possession of knowledge on the Internet with registered keys", i.e., the subtleties of concurrent knowledge-extraction in the public-key model. We note that this attack is of a *man-in-the-middle* nature, and is related to malleability of protocols.

Now, we proceed to formulate concurrent verifier security in light of the attack against the protocol of [7,23]. The security notion assuring that a malicious prover P^* does "know" what it claims to know, when it is concurrently interacting with the honest verifier V, can informally be formulated as: for any x, if P^* can convince V (with public-key PK) of "$x \in L$" (for an $\mathcal{NP}$-language L) by concurrent interactions, then there exists a PPT knowledge-extractor that outputs a witness for $x \in L$. This is a natural extension of the normal arguments of knowledge into the concurrent public-key setting. However, this formulation approach is problematic in the concurrent public-key setting. The reason is: the statements being proved may be related to PK, and thus the extracted witness may be related to the corresponding secret-key SK (even just the secret-key as shown by the concrete attack on the protocol of [7,23]); But, in knowledge-extraction the PPT extractor may have already possessed SK. To solve this subtlety, we require the extracted witness, together with adversary's view, to be *independent* of SK. But, the problem here is how to formalize such independence, in particular, w.r.t. a CMIM? We solve this in the spirit of non-malleability formulation [9]. That is, we consider the message space (distribution) of SK, and such independence is roughly formulated as follows: let SK be the secret-key and SK' is an element randomly and independently distributed over the space of SK, then we require that, for any polynomial-time computable relation R, the probability $\Pr[R(\bar{w}, SK, view) = 1]$

is negligibly close to $\Pr[R(\bar{w}, SK', view) = 1]$, where $\bar{w}$ is the set of witnesses extracted by the knowledge extractor for successful concurrent sessions and $view$ is the view of P^*. This captures the intuition that P^* does, in fact, "know" the witnesses to the statements whose validations are successfully conveyed by concurrent interactions.

Definition 1 (concurrent knowledge-extraction (CKE) in the public-key model) *We say that a protocol $\langle P, V\rangle$ is* concurrently knowledge-extractable *in the BPK model w.r.t. some admissible language set $\mathcal{L}$ and some key-validating relation R_{KEY}, if for any positive polynomial $s(\cdot)$, any s-concurrent malicious prover P^*, there exist a pair of (expected) polynomial-time algorithms S (the simulator) and E (the extractor) such that for any sufficiently large n, any auxiliary input $z \in \{0,1\}^*$, and any polynomial-time computable relation R (with components drawn from $\{0,1\}^* \cup \{\perp\}$), the following hold in accordance with the experiment $\mathbf{Expt}_{\mathbf{CKE}}(1^n, z)$ described below:*

$\mathbf{Expt}_{\mathbf{CKE}}(1^n, z)$

The simulator $S = (S_{KEY}, S_{PROOF})$:
$(PK, SK, SK') \longleftarrow S_{KEY}(1^n)$, where the distribution of (PK, SK) is identical with that of the output of the key-generation stage of the honest verifier V, $R_{KEY}(PK, SK) = R_{KEY}(PK, SK') = 1$ and the distributions of SK and SK' are identical and *independent*. In other words, SK and SK' are two random and independent secret-keys corresponding to PK.

$(str, sta) \longleftarrow S_{PROOF}^{P^*(1^n,\, PK,\, z)}(1^n, PK, SK, z)$. That is, on inputs $(1^n, PK, SK, z)$ and with oracle access to $P^*(1^n, PK, z)$ (by providing random tape to P^* and running P^* as subroutine), the simulator S outputs a simulated transcript str, and some state information sta to be transformed to the knowledge-extractor E.

We denote by $S_1(1^n, z)$ the random variable str (in accordance with above processes of S_{KEY} and S_{PROOF}). For any $(PK, SK) \in R_{KEY}$ and any $z \in \{0,1\}^*$, we denote by $S_1(1^n, PK, SK, z)$ the random variable describing the first output of $S_{PROOF}^{P^*(1^n,\, PK,\, z)}(1^n, PK, SK, z)$ (i.e., str specific to (PK, SK)).

The knowledge-extractor E:
$\overline{w} \longleftarrow E(1^n, sta, str)$. On (sta, str), E outputs a list of witnesses to statements whose validations are successfully conveyed in str.

- **Simulatability.** *The following ensembles are indistinguishable: $\{view_{P^*}^{V(SK)}(1^n, z, PK)\}_{(PK,SK)\in R_{KEY}, z\in\{0,1\}^*}$ and $\{S_1(1^n, PK, SK, z)\}_{(PK,SK)\in R_{KEY}, z\in\{0,1\}^*}$.*
- **Secret-key independent knowledge-extraction.** *E, on inputs $(1^n, str, sta)$, outputs witnesses to all statements successfully proved in accepting sessions in str. Specifically, E outputs a list of strings $\overline{w} = (w_1, w_2, \cdots, w_{s(n)})$, satisfying the following:*
  - *w_i is set to be $\perp$, if the i-th session in str is not accepting (due to abortion or verifier verification failure), where $1 \le i \le s(n)$.*
  - Correct knowledge-extraction (for individual statements): *In any other cases (i.e., for successful sessions), with overwhelming probability $(x_i, w_i) \in R_L$, where x_i is the common input selected by P^* for the i-th session in str and R_L is the admissible $\mathcal{NP}$-relation for $L \in \mathcal{L}$ set by P^* in str.*

- (Joint) knowledge extraction independence (KEI): $\Pr[R(SK, \overline{w}, str) = 1]$ *is negligibly close to* $\Pr[R(SK', \overline{w}, str) = 1]$.

The probabilities are taken over the randomness of S *in the key-generation stage (i.e., the randomness for generating* (PK, SK, SK')*) and in all proof stages, the randomness of* E*, and the randomness of* P^**. If the KEI property holds for any (not necessarily polynomial-time computable) relation* R*, we say the protocol* $\langle P, V \rangle$ *satisfies* statistical *CKE.*

We first note that the above CKE formulation follows the simulation-extraction approach of [19]. Here, the key augmentation, besides some other adaptations in the public-key model, is the property of knowledge-extraction independence (KEI) explicitly required (the KEI notion originally appeared in the incomplete work of [23], August 2006 update). Though the CKE and KEI notions are formulated in the framework of public-key model, they are actually applicable to protocols in the plain model, in general, in order to capture knowledge extractability against concurrent adversaries interacting with honest players of secret values.

Below, we discuss and clarify various subtleties surrounding the CKE formulation. More details are referred to the full paper [20].

Simulated public-keys vs. real public-keys. In our CKE formulation, the simulation-extraction is w.r.t. *simulated* public-keys. A natural and intuitive strengthening of the CKE formulation might be: the simulator/extractor uses the *same* public-keys of the honest verifiers. In this case, as the simulator/extractor does not possess honest verifier's secret-key, the KEI property can be waived. But, the observation here is: constant-round CKE (*whether ZK or not*) with real public-keys are impossible. Specifically, constant-round CKE with real public-keys implies constant-round CZK (potentially, concurrent non-malleable ZKPOK) *in the plain model* by viewing verifier's public-keys as a part of common inputs, which is however impossible at least in the black-box sense [5].

On CKE with independent language. With the above KEI formulation, we are actually formulating the independence of the witnesses, used ("*known*") by CMIM adversary, on the secret-key (witness) used by verifier (who may in turn play the role of prover in some sub-protocols). A naive solution for KEI, which appears to make sense, may be to require the language and statements being proved are independent of verifier's public-keys. But, this approach has the following problems: Firstly, if the protocol is for $\mathcal{NP}$-Complete, the statements being proved, selected adaptively by the adversary, can always be related to verifier's public-key (e.g., via $\mathcal{NP}$-reductions); Secondly, as the statements being proved are selected adaptively by the CMIM adversary on the fly, in general it is hard to distinguish whether the maliciously chosen statements are independent of verifiers' public-keys or not; Thirdly, the applicability of this approach is significantly limited (and even useless in practice, where keys are used in essential ways in malicious settings like the Internet).

CKE vs. concurrent soundness. As a consequence of the attack on the CZK protocol of [7,23] that is both concurrently sound and can be implemented based on any OWF, we show that, assuming any OWF, CKE is a strictly stronger notion for concurrent verifier security than concurrent soundness in the public-key model. We note that,

prior to our work, whether A/POK is strictly stronger than soundness (in the concurrent public-key setting) is unknown.

Taking adversary's view, i.e., str, into account for capturing KEI. We note this is necessary for the completeness of KEI formulation. Specifically, consider the following (seemingly impossible) case that: for any extracted w_i in $\bar{w}$, $w_i = PRF_s(SK)$, where the seed s could be either a part of the adversary's random tape or a value computed from its view. In other words, the witnesses extracted are always dependent on the secret-key used by the simulator/extracotor, and thus the adversary may not necessarily be aware of the extracted knowledge. But, without taking account of adversary's view, $\Pr[R(SK, \bar{w}) = 1]$ is still negligibly close to $Pr[R(SK', \bar{w}) = 1]$ in this case for any polynomial-time computable relation R.

We note that, explicitly taking account of adversary's view seems to be necessary for correct and complete CNM formulations, whenever (not necessarily extractable) knowledge independence is a necessary property to be considered. We note that this issue is applicable to some related works, and can also be traced back to the origin of NM formulation [9].

On extending the Bellare-Goldreich (BG) quantitative approach for stand-alone POK into the concurrent setting. We note that, besides the subtle KEI issue, there are some difficulties (or inconveniences) to extend the BG quantitative approach for stand-alone POK [1,11,2] (i.e., the quantitative definition of expected knowledge-extraction time that is in inverse proportion to the probability the adversary convinces of the statement) into the concurrent setting. Below, we consider two possible approaches to extend the BG quantitative approach (for stand-alone POK) into the concurrent setting.

The first approach is: for each of all the concurrent sessions, we consider the probability that the adversary (i.e., the malicious prover P^*) successfully finishes the session. Denote by p_i the probability that the adversary successfully finish the i-th session. Note that this probability is particularly taken over the random coins of P^* and all random coins of the honest verifier instances in all concurrent sessions. But, within the simulation-extraction formulation framework, it is difficult to give a precise quantitative definition of the knowledge-extraction time inversely proportional to p_i. The reason is: when we apply the underlying stand-alone knowledge-extractor (guaranteed by the Bellare-Goldreich POK definition) on the successful i-th session in the simulated transcript, the knowledge-extraction is actually with respect to the probability, denote p'_i, that P^* successfully finish the i-th session when the coins of the honest verifier instances in all other sessions (other than the i-th session) are fixed (i.e., determined by the simulated transcript str). Clearly, p'_i can be totally different from p_i (e.g., p_i may be non-negligible, but p'_i can be negligible), and thus the knowledge-extraction time w.r.t p'_i can be totally different from that w.r.t p_i.

The second approach is to separate the simulation and knowledge-extraction. Specifically, besides indistinguishable simulation, we *separately* require (regardless of the simulated transcript) that for any x *selected adaptively by the adversary during its concurrent attack*, if the adversary P^* can, with probability p_x, convince the honest verifier of the statement "$x \in L$" in one of the $s(n)$ sessions by concurrent interactions, the knowledge-extraction time should be in inverse proportion to p_x. We note that this

approach does not work. On the one hand, suppose P^* convinces $x \in L$ in one of the $s(n)$ sessions (say the i-th session) with some non-negligible probability, but with negligible probability in all other sessions. In this case, it is okey if the knowledge extraction is w.r.t. the i-th session, but will fail w.r.t other sessions. On the other hand, one may argue that to remedy the above subtlety, we can add a (polynomial-time) bound on the knowledge-extraction in each session, but this solution fails if the adversary convinces of the statement "$x \in L$" with negligible probability in all sessions. In general, it may be hard to distinguish the two cases, i.e., the case that P^* succeeds with negligible probability in all sessions and the case that P^* may succeed with non-negligible probability in some (but not all) sessions.

We note that the work [8] takes the approach of extending the BG (stand-alone) POK formulation into the concurrent setting in the BPK model, *without clarifying the above subtleties*. For example, the running time of the knowledge-extractor E formulated in [8] is w.r.t. the probability p_i, but it is unclear how to handle the issue of p_i' versus p_i as clarified above. In addition, the work [8] does not capture adaptive language selection by the concurrent malicious prover, and does not capture the KEI issue (it is unclear how about if the knowledge extracted by E is dependent on verifier's secret-key that is actually generated by E itself). As a consequence, the formulation approach of [8] may be less convenient to use (particularly for analyzing complex cryptographic protocols running concurrently with public-keys). To our knowledge, still no formal proofs in accordance with the formulation approach of [8] are presented in existing works. In comparison, we suggest our CKE formulation is of conceptual clarity and simplicity, is easier to work with and can be efficiently achievable, and is well compatibility of the normal simulation/extraction formulation approach for concurrent security of protocols. We also remind that our CKE formulation implicitly assumes that verifier's public-key corresponds to multiple secret-keys (in the sense that protocols with unique secret-key for the verifier may trivially *not* satisfy the CKE security), which however can typically be achieved with the common key-pair trick [18]. In general, cryptography literature should welcome diversified approaches for modeling and achieving security goals of cryptographic systems, particularly witnessed by the evolution history of public-key encryption.

4 Overview of Achieving CZK-CKE in the BPK Model

In this section, we present the high-level overview of achieving constant-round CZK-CKE arguments in the BPK model, with details referred to the full paper [20].

The starting point is the basic and central Feige-Shamir ZK (FSZK) structure [10]. The FSZK structure is conceptually simple and is composed of two WIPOK sub-protocols. In more details, let f be a OWF, in the first WIPOK sub-protocol with the verifier V serving as the knowledge-prover, V computes $(y_0 = f(s_0), y_1 = f(s_1))$ for randomly chosen s_0 and s_1; then V proves to the prover P the knowledge of the preimage of either y_0 or y_1. In the second WIPOK sub-protocol with P serving as the knowledge-prover for an $\mathcal{NP}$-language L, on common input x, P proves to V the knowledge of either a valid $\mathcal{NP}$-witness w for $x \in L$ or the preimage of either y_0 or y_1. FSZK is also argument of knowledge, and can be instantiated practically (without going through general $\mathcal{NP}$-reductions) by the Σ_{OR} technique [6,23].

Let (y_0, y_1) serve as the public-key of V and s_b (for a random bit b) as the secret-key, the public-key version of FSZK is CZK in the BPK model [23]. But, we show that the public-key version of FSZK is not of concurrently soundness [21], needless to say concurrent knowledge-extractability (indeed, FSZK was not designed for the public-key model). We hope to add the CKE property to FSZK in the BPK model (and thus get concurrent security both for the prover and for the verifier simultaneously), while maintaining its conceptual simplicity and its suitability to be instantiated practically.

The subtle point is: we are actually dealing with a CMIM attacker who manages to malleate, in a malicious and unpredictable way, the public-keys and knowledge-proof interactions of the verifier in one session into the statements and knowledge-proof interactions in another concurrent session. To add CKE security to FSZK in the BPK model, some non-malleable tools seem to be required. Here, we show how to do so without employing such tools.

The crucial idea behind achieving our goal is to strengthen the first sub-protocol to be *statistical* WIPOK, and require the prover to first, before starting the second WIPOK sub-protocol, commit to the supposed witness to c_w by running a *statistically-binding* commitment scheme. This guarantees that if the witness committed to c_w is dependent on the secret-key used by V, there are, in fact, certain differences between the interaction distribution when V uses $SK = s_0$ and the one when V uses $SK = s_1$. We can, in turn, use such distribution differences to violate the statistical WI of the first sub-protocol, which then implies *statistical* CKE. This solution, however, loses CZK in general, since the second WI sub-protocol is run w.r.t. commitments to different values in real interactions and in the simulation. To deal with this problem we employ a stronger second sub-protocol, i.e., strong WI argument/proof-of-knowledge (strong WIPOK) [11]. Note that composing the commitment and the SWI yields a regular WI, and thus the CZK property is salvaged.

Employing SWI complicates the protocol structure, and incurs protocol inefficiency. It is, therefore, desirable to still use a regular WIPOK in the second sub-protocol, for conceptual simplicity and efficiency. To bypass the subtleties of SWI for the CZK proof, we employ a double-commitments technique. Specifically, we require the prover to produce a *double* of statistically-binding commitments, c_w and c_{sk}, before starting the second WIPOK sub-protocol of FSZK, where c_w is supposed to commit to a valid $\mathcal{NP}$-witness for $x \in L$ and c_{sk} is supposed to commit to the preimage of either y_0 or y_1. Double commitments can bypass, by hybrid arguments, the subtleties of SWI for the CZK proof. But, the provable CKE property with double commitments turns out to be much subtler. Specifically, due to the double commitments used, the value extracted can be either the value committed to c_w or that to c_{sk}. If it is ensured that the valued extracted is always the one committed to c_w (i.e., satisfying the correct knowledge-extraction property of Definition 1), we can get statistical CKE in the same way as the SWI-based solution. By the one-wayness of f, the value extracted in polynomial time cannot be the preimage of y_{1-b} (recall the secret-key is s_b). But, how about the possibility that the value extracted is just the secret-key s_b committed to c_{sk}? Consider the following adversarial strategy:

With non-negligible probability p, P^* commits s_0 (resp., s_1) to c_{sk} in a session (possibly by malleating verifier's public-key into c_{sk}); Then, *possibly by malleating the first WIPOK sub-protocol concurrent interactions*, P^* successfully finishes the second

WIPOK sub-protocol of the session with s_0 (resp., s_1) as the witness, in case the verifier V uses s_0 (resp., s_1) as the secret-key; However, with the same probability p, P^* commits both a valid witness w to c_w and s_0 (resp. s_1) to c_{sk}, and then successfully finishes the second WIPOK sub-protocol with w as the witness in case V uses s_1 (resp., s_0) as secret-key. Note that, for this adversarial strategy, with non-negligible probability p the value extracted will just be the secret-key (that is also used by the extractor itself). But, we do not know how to reach contradiction under standard polynomial assumptions in this case. In particular, this adversarial strategy does not violate the statistical WI of the first WIPOK sub-protocol: with probability $2p$, the value committed to c_{sk} is s_σ for *both* $\sigma \in \{0,1\}$, no matter which secret-key is used by the verifier.

To overcome this technical difficulty, we employ complexity leveraging in a novel way. Specifically, on the system parameter n, we assume the OWF f is hard against sub-exponential 2^{n^c}-time adversaries for some constant c, $0 < c < 1$. But, the commitment c_{sk} is generated on a relatively smaller security parameter n_{sk} such that n_{sk} and n are polynomially related (i.e., any quantity that is a polynomial of n is also another polynomial of n_{sk}) but $poly(n) \cdot 2^{n_{sk}} \ll 2^{n^c}$. This complexity leveraging ensures that, with at most negligible probability, the value extracted can be the secret-key s_b committed to c_{sk}, from which the correctness of knowledge-extraction (and then the *statistical* CKE security) is established. The reasoning is as follows: For any i, $1 \leq i \leq s(n)$, suppose with non-negligible probability p an s-concurrent malicious P^* can successfully finish the i-th session with c_{sk} committing to s_σ, $\sigma \in \{0,1\}$, when the honest verifier (and also the extractor) uses s_σ as its secret-key; Then, by the *statistical* WI property of the first WIPOK sub-protocol, with the same probability p, P^* successfully finishes the i-th session with c_{sk} committing to s_σ, when the honest verifier uses $s_{1-\sigma}$ as the secret-key. In the later case, we can open c_{sk} to get s_σ by brute force in $poly(n) \cdot 2^{n_{sk}}$-time, which however violates the sub-exponential hardness of y_σ because $poly(n) \cdot 2^{n_{sk}} \ll 2^{n^c}$.

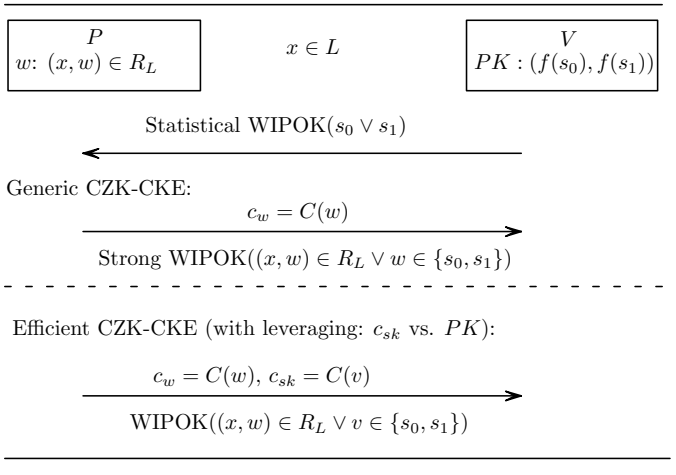


Fig. 1. Depiction of CZK-CKE from FSZK

We stress that complexity leveraging via the sub-exponential hardness assumption on verifier's public-key is only for provable security analysis to frustrate concurrent man-in-the-middle. Both CZK simulation and CKE knowledge-extraction are still in polynomial-time. We suggest that the use of complexity leveraging for frustrating CMIM could be a useful paradigm, different from the uses of complexity leveraging in existing works for protocols in the BPK model (e.g., [4]). The complexity-leveraging based efficient and conceptually simple CZK-CKE solution can be further practically instantiated for some common number-theoretic languages.

The CZK-CKE protocols from FSZK are roughly depicted in Figure 1. We also show that all other FSZK possible component variants within the given protocol structure of Figure 1, are essentially not provably (black-box) CZK-CKE secure in the BPK model, which is, perhaps, somewhat puzzling.

Acknowledgment. Yunlei Zhao thanks Giovanni Di Crescenzo, Ivan Visconti and Frances F. Yao for helpful discussions.

References

1. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
2. Bellare, M., Goldreich, O.: On Probabilistic versus Deterministic Provers in the Definition of Proofs of Knowledge. Cryptology ePrint Archive, Report 2006/359
3. Blum, M.: How to Prove a Theorem so No One Else can Claim It. In: Proceedings of the International Congress of Mathematicians, pp. 1444–1451 (1986)
4. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero-Knowledge. In: ACM Symposium on Theory of Computing, pp. 235–244 (2000)
5. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-Box Concurrent Zero-Knowledge Requires (Almost) Logarithmically Many Rounds. SIAM Journal on Computing 32(1), 1–47 (2002)
6. Cramer, R., Damgard, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
7. Di Crescenzo, G., Visconti, I.: Concurrent Zero-Knowledge in the Public-Key Model. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 816–827. Springer, Heidelberg (2005)
8. Di Crescenzo, G., Visconti, I.: On Defining Proofs of Knowledge in the Bare Public-Key Model. In: ICTCS (2007)
9. Dolev, D., Dwork, C., Naor, M.: Non-Malleable Cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)
10. Feige, U., Shamir, A.: Zero Knowledge Proofs of Knowledge in Two Rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–544. Springer, Heidelberg (1990)
11. Goldreich, O.: Foundation of Cryptography-Basic Tools (2001)
12. Goldreich, O.: Foundations of Cryptography-Basic Applications (2002)
13. Goldreich, O., Micali, S., Wigderson, A.: Proofs that Yield Nothing But Their Validity or All languages in $\mathcal{NP}$ Have Zero-Knowledge Proof Systems. JACM 38(1), 691–729 (1991)
14. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. In: ACM Symposium on Theory of Computing, pp. 291–304 (1985)

15. Halevi, S., Micali, S.: Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 201–215. Springer, Heidelberg (1996)
16. Micali, S., Reyzin, L.: Soundness in the Public-Key Model. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 542–565. Springer, Heidelberg (2001)
17. Naor, M.: Bit Commitment Using Pseudorandomness. Journal of Cryptology 4(2), 151–158 (1991)
18. Naor, M., Yung, M.: Public-Key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. In: STOC 1990, pp. 427–437 (1990)
19. Pass, R., Rosen, A.: New and Improved Constructions of Non-Malleable Cryptographic Protocols. SIAM Journal on Computing 38(2), 702–752 (2008)
20. Yao, A.C., Yung, M., Zhao, Y.: Concurrent Knowledge Extraction in the Public-Key Model. ECCC, Report 2007/002 (2007); Available also from Cryptology ePrint Archive, Report 2010/
21. Yung, M., Zhao, Y.: Interactive Zero-Knowledge with Restricted Random Oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 21–40. Springer, Heidelberg (2006)
22. Zhao, Y., Nielsen, J.B., Deng, R., Feng, D.: Generic yet Practical ZK Arguments from any Public-Coin HVZK. ECCC, 2005/162 (2005)
23. Zhao, Y.: Concurrent/Resettable Zero-Knowledge With Concurrent Soundness in the Bare Public-Key Model and Its Applications. Cryptology ePrint Archive, Report 2003/265 (2003)

On the k-Independence Required by Linear Probing and Minwise Independence

Mihai Pătraşcu and Mikkel Thorup

AT&T Labs

Abstract. We show that linear probing requires 5-independent hash functions for expected constant-time performance, matching an upper bound of [Pagh et al. STOC'07]. For $(1+\varepsilon)$-approximate minwise independence, we show that $\Omega(\lg \frac{1}{\varepsilon})$-independent hash functions are required, matching an upper bound of [Indyk, SODA'99]. We also show that the multiply-shift scheme of Dietzfelbinger, most commonly used in practice, fails badly in both applications.

1 Introduction

The concept of k-wise independence was introduced by Wegman and Carter [19] in FOCS'79 and has been the cornerstone of our understanding of hash functions ever since. Formally, a family $\mathcal{H} = \{h : [u] \to [b]\}$ of hash functions is k-independent if (1) for any distinct keys $x_1, \ldots, x_k \in [u]$, the hash codes $h(x_1), \ldots, h(x_k)$ are independent random variables; and (2) for any fixed x, $h(x)$ is uniformly distributed in $[b]$.

As the concept of independence is fundamental to probabilistic analysis, k-independent functions are both natural and powerful in algorithm analysis. They allow us to replace the heuristic assumption of truly random hash functions with real (implementable) hash functions that are still "independent enough" to yield provable performance guarantees. We are then left with the natural goal of understanding the independence required by algorithms.

The canonical construction of a k-independent family is based on polynomials of degree $k-1$. Let $p \geq u$ be prime. Picking random $a_0, \ldots, a_{k-1} \in \{0, \ldots, p-1\}$, the hash function is defined by:

$$h(x) = \Big(\big(a_{k-1} x^{k-1} + \cdots + a_1 x + a_0 \big) \bmod p \Big) \bmod b$$

For $p \gg b$, the hash function is statistically close to k-independent.

In simple cases, 2-independence suffices. For instance, if one implements a hash table by chaining, the time it takes to query x is proportional to the number of keys y colliding with x (i.e. $h(x) = h(y)$). Thus, pairwise independence of $h(x)$ and $h(y)$ is all we need to understand the expected query time.

At the other end of the spectrum, $O(\lg n)$-independence suffices in a vast majority of applications. One reason for this is the Chernoff bounds of [14] for k-independent events, whose probability bounds differ from the full-independence

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 715–726, 2010.

Chernoff bound by $2^{-\Omega(k)}$. Another reason is that random graphs with $O(\lg n)$-independent edges [1] share many of the properties of truly random graphs.

In this paper, we study two compelling applications in which independence $2 < k < \lg n$ is currently needed: linear probing and minwise-independent hashing. (The reader unfamiliar with these applications will find more details below.) For linear probing, Pagh et al. [10] showed that 5-independence suffices, thus giving the first realistic implementation of linear probing with formal guarantees. For minwise-independence, Indyk [8] showed that ε approximation can be obtained using $O(\lg \frac{1}{\varepsilon})$-independence.

In both cases, it was known that 2-independence does not suffice [10,3], and, indeed, the simplest family $x \mapsto (ax + b) \bmod p$ provides a counterexample. However, a significant gap remained to the upper bounds.

In this paper, we close this gap, showing that both upper bounds are, in fact, tight. We do this by exhibiting carefully constructed families for which these algorithms fail: for linear probing, we give a 4-independent family that leads to $\Omega(\lg n)$ query time; and for minwise independence, we give an $\Omega(\lg \frac{1}{\varepsilon})$-independent family that leads to 2ε approximation.

Concrete schemes. Our results give a powerful understanding of a natural combinatorial resource (independence) for two important algorithmic questions. In other words, they are limits on how far the *paradigm* of independence can bring us. Note, however, that independence is only one property that concrete hash schemes have. In a particular application, a hash scheme can behave much better that its independence guarantees, if it has some other probabilistic property unrelated to independence.

In practice, the most popular hash function is not $x \mapsto \big((ax+b) \bmod p\big) \bmod u$, but Dietzfelbinger's multiply-shift scheme [6], which can be twice as fast [16]. To hash w-bit integers to the range $b = 2^\ell$, the scheme picks a random a of $2w$ bits, and computes $(ax) >> (2w - \ell)$, where $>>$ denotes unsigned shift.

In the full version of this paper, we prove that linear probing with multiply-shift hashing suffers from $\Omega(\lg n)$ expected running times. Similarly, minwise independent hashing has a very large approximation, of $\varepsilon = \Omega(\lg n)$. While these results are not surprising, given the "moral similarity" of multiply-shift and $ax+b$ schemes, they do require rather involved arguments. We feel this effort is justified, as it brings the theoretical lower bounds in line with programming reality.

In the same vein, one could ask to replace our lower bounds, which construct artificial families of high independence, by a proof that the family of polynomials fails. While this is an intriguing algebraic question, we do note that it is far less pressing from a practical viewpoint. Even with the best known implementation tricks (such as computing modulo Mersenne primes), higher degree polynomials make rather slow hash functions, and are not widely used. For instance, the fastest known 5-independent family is tabulation based [17,18], and it outperforms polynomials by a factor of 5 or more. In a recent manuscript, we show [12] that tabulation-based hash functions do, in fact, behave better than their independence would suggest: for instance, 3-independent tabulation yields $O(1)$ running times for linear probing.

1.1 Technical Discussion: Linear Probing

Linear probing uses a hash function to map a set of keys into an array of size b. When inserting x, if the desired location $h(x)$ is already occupied, the algorithm scans $h(x)+1, h(x)+2, \ldots$ until an empty location is found, and places x there. The query algorithm starts at $h(x)$ and scans either until it finds x, or runs into an empty position, which certifies that x is not in the hash table. We assume constant load of the hash table, e.g. the number of keys is $n \leq \frac{2}{3}b$.

This classic data structure is the most popular implementation of hash tables, due to its unmatched simplicity and efficiency. On modern architectures, access to memory is done in cache lines (of much more than one element), so inspecting a few consecutive values typically translates into just one memory probe. Even if the scan straddles a cache line, the behavior will still be better than a second random memory access on architectures with prefetching. Empirical evaluations [2,7,11] confirm the practical advantage of linear probing over other known schemes, while cautioning [7,18] that it behaves quite unreliably with weak hash functions (such as 2-independent). Taken together, these findings form a strong motivation for theoretical analysis.

Linear probing was first shown to take expected constant time per operation in 1963 by Knuth [9], in a report now considered the birth of algorithm analysis. However, this required truly random hash functions.

A central open question of Wegman and Carter [19] was how linear probing behaves with k-independence. Siegel and Schmidt [13,15] showed that $O(\lg n)$-independence suffices. Recently, Pagh et al. [10] showed that even 5-independent hashing works. We now close this line of work, showing that 4-independence is not enough.

Review of the 5-independence upper bound. To better situate our lower bounds, we begin by reviewing the upper bound of [10]. The main probabilistic tool featuring in this analysis is a 4^{th} moment bound. Consider throwing n balls into b bins uniformly. Let X_i be the probability that ball i lands in the first bin, and $X = \sum_{i=1}^{n} X_i$ the number of balls in the first bin. We have $\mu = \mathbf{E}[X] = \frac{n}{b}$. Then, the k^{th} moment of X is defined as $\mathbf{E}[(X-\mu)^k]$.

As long as our placement of the balls is k-independent, the k^{th} moment is identical to the case of full independence. For instance, the 4^{th} moment is:

$$\mathbf{E}[(X-\mu)^4] = \mathbf{E}\big[\big(\sum_i (X_i - \tfrac{1}{b})\big)^4\big] = \sum_{i,j,k,l} \mathbf{E}\big[(X_i - \tfrac{1}{b})(X_j - \tfrac{1}{b})(X_k - \tfrac{1}{b})(X_l - \tfrac{1}{b})\big].$$

The only question in calculating this quantity is the independence of sets of at most 4 items. Thus, 4-independence preserves the 4^{th} moment of full randomness.

Moments are a standard approach for bounding the probability of large deviations. Let's say that we expect μ items in the bin, but have capacity 2μ; what is the probability of overflow? A direct calculation shows that the 4^{th} moment is $\mathbf{E}[(X-\mu)^4] = O(\mu^2)$. Then, by a Markov bound, the probability of overflow is $\Pr[X \geq 2\mu] = \Pr[(X-\mu)^4 \geq \mu^4] = O(1/\mu^2)$. By contrast, if we only have 2-independence, we can use the 2^{nd} moment $\mathbf{E}[(X-\mu)^2] = O(\mu)$ and

obtain $\Pr[X \geq 2\mu] = O(1/\mu)$. Observe that the 3rd moment is not useful for this approach, since $(X-\mu)^3$ can be negative, so Markov does not apply.

To apply moments to linear probing, we consider a perfect binary tree spanning the array. For notational convenience, let us assume that the load is at most $n \leq b/3$. A node on level ℓ has 2^ℓ array positions under it, and we expect $2^\ell/3$ keys to be hashed to one of them (but more or less keys may actually appear in the subtree, since items are not always placed at their hash position). Call the node dangerous if at least $\frac{2}{3}2^\ell$ keys hash to it.

In the first stage, we will bound the total time it takes to construct the hash table (the cost of inserting n distinct items). If the table consists of runs of $k_1, k_2, \ldots$ elements ($\sum k_i = n$), the cost of constructing it is bounded from above by $O(k_1^2 + k_2^2 + \ldots)$. To bound these runs, we make the following crucial observation: if a run contains between 2^ℓ and $2^{\ell+1}$ elements, then some node at level $\ell - 2$ above it is dangerous.

For a proof, assume the run goes from positions i to j. The interval $[i, j]$ is spanned by 4 to 9 nodes on level $\ell - 2$. Assume for contradiction that none are dangerous. The first node, which is not completely contained in the interval, contributes less than $\frac{2}{3}2^{\ell-2}$ elements to the run (it the most extreme case, this many elements hashed to the last location of that node). But the subsequent nodes all have more than $2^{\ell-2}/3$ free locations in their subtree, so 2 more nodes absorb all excess elements. Thus, the run cannot go on for 4 nodes, contradiction.

This observation gives an upper bound on the cost: add $O(2^{2\ell})$ for each dangerous node at some level ℓ. Denoting by $p(\ell)$ the probability that a node on level ℓ is dangerous, the expected cost is thus $\sum_\ell (n/2^\ell) \cdot p(\ell) \cdot 2^{2\ell} = \sum_\ell n \cdot 2^\ell p(\ell)$. Using the 2nd moment to bound $p(\ell)$, one would obtain $p(\ell) = O(2^{-\ell})$, so the total cost would be $O(n \lg n)$. However, the 4th moment gives $p(\ell) = O(2^{-2\ell})$, so the cost at level ℓ is now $O(n/2^\ell)$. In other words, the series starts to decay geometrically and is bounded by $O(n)$.

To bound the running time of one particular operation (query or insert q), we actually use the stronger guarantee of 5-independence. If the query lands at a uniform place conditioned on everything else in the table, then at each level it will pay the "average cost" of $O(1/2^\ell)$, which sums up to $O(1)$.

Our results. Two intriguing questions pop out of this analysis. First, is the independence of the query really crucial? Perhaps one could argue that the query behaves like an average operation, even if it is not completely independent of everything else. Secondly, one has to wonder whether 3-independence suffices (by using something other than 3rd moment): all that is needed is a bound slightly stronger than 2nd moment in order to make the series decay geometrically!

We answer both questions in strong negative terms. The complete understanding of linear probing with low independence is summarized in Table 1. Addressing the first question, we show that 4-independence cannot give expected time per operation better than $\Omega(\lg n)$, even though n operations take $O(n)$. Our proof demonstrates an important phenomenon: even though most bins have low load, a particular element's hash code could be correlated with the (uniformly random) choice of *which* bins have high load.

Table 1. Expected time bounds with a bad family of k-independent hash functions. Construction time refers to the total time over n different insertions.

Independence	2	3	4	≥ 5
Query time	$\Theta(\sqrt{n})$	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(1)$
Construction time	$\Theta(n \lg n)$	$\Theta(n \lg n)$	$\Theta(n)$	$\Theta(n)$

An even more striking illustration of this fact happens for 2-independence: the query time blows up to $\Omega(\sqrt{n})$ in expectation, since we are left with no independence at all after conditioning on the query's hash. This demonstrates a very large separation between linear probing and collision chaining, which enjoys $O(1)$ query times even for 2-independent hash functions.

Addressing the second question, we show that 3-independence is not enough to guarantee even a construction time of $O(n)$. Thus, in some sense, the 4^{th} moment analysis is the best one can hope for.

1.2 Technical Discussion: Minwise Independence

This concept was introduced by two classic algorithms: detecting near-duplicate documents [3,4] and approximating the size of the transitive closure [5]. The basic step in these algorithms is estimating the size of the intersection of pairs of sets, relative to their union: for A and B, we want to find $\frac{|A \cap B|}{|A \cup B|}$ (the *Jaccard similarity coefficient*). To do this efficiently, one can choose a hash function h and maintain $\min h(A)$ as the sketch of an entire set A. If the hash function is truly random, we have $\Pr[\min h(A) = \min h(B)] = \frac{|A \cap B|}{|A \cup B|}$. Thus, by repeating with several hash functions, or by keeping the bottom k elements with one hash function, the Jaccard coefficient can be estimated up to a small approximation.

To make this idea work, the property that is required of the hash function is *minwise independence*. Formally, a family of functions $\mathcal{H} = \{h : [u] \to [u]\}$ is said to be minwise independent if, for any set $S \subset [u]$ and any $x \notin S$, we have $\Pr_{h \in \mathcal{H}}[h(x) < \min h(S)] = \frac{1}{|S|+1}$. In other words, x is the minimum of $S \cup \{x\}$ only with its "fair" probability $\frac{1}{|S|+1}$.

As good implementations of exact minwise independent functions are not known, the definition is relaxed to ε-minwise independent, where $\Pr_{h \in \mathcal{H}}[h(x) < \min h(S)] = \frac{1 \pm \varepsilon}{|S|+1}$. Using such a function, we will have $\Pr[\min h(A) = \min h(B)] = (1 \pm \varepsilon)\frac{|A \cap B|}{|A \cup B|}$. Thus, the ε parameter of the minwise family dictates the best approximation achievable in the algorithms (which *cannot* be improved by repetition).

Indyk [8] gave the only implementation of minwise independence with provable guarantees, showing that $O(\lg \frac{1}{\varepsilon})$-independent functions are ε-minwise independent.

His proof uses another tool enabled by k-independence: the inclusion-exclusion principle. Say we want to bound the probability that at least one of n events is "good." We can define $p(k) = \sum_{S \subset [n], |S|=k} \Pr[\text{all } S \text{ are good}]$. Then, the

probability that at least one event is good is, by inclusion-exclusion, $p(1) - p(2) + p(3) - p(4) + \ldots$. If we only have k-independence (k odd), we can upper bound the series by $p(1) - p(2) + \cdots + O(p(k))$. In the common scenario that $p(k)$ decays exponentially with k, the trimmed series will only differ from the full independence case by $2^{-\Omega(k)}$. Thus, k-independence achieves bounds exponentially close to full independence, whenever probabilities can be computed by inclusion-exclusion. This turns out to be the case for minwise independence: we can express the probability that at least some element in S is below x by inclusion-exclusion.

In this paper, we show that, for any $\varepsilon > 0$, there exist $\Omega(\lg \frac{1}{\varepsilon})$-independent hash functions that are no better than (2ε)-minwise independence. Thus, ε-minwise independence requires $\Omega(\lg \frac{1}{\varepsilon})$ independence.

2 Time $\Theta(\sqrt{n})$ with 2-Independence

We define a 2-independent hash family such that the expected query time is $\Theta(\sqrt{n})$. The main idea of the proof is that the query can play a special role: even if most portions of the hash table are lightly loaded, the query can be correlated with the portions that *are* loaded. We assume that b is a power of two, and we store $n = b/2$ keys. We also assume that $\sqrt{n}$ is a power of two.

We can think of the stored keys and the query key as fixed, and we want to find bad ways of distributing them 2-independently into the range $[b]$. To extend the hash function to the entire universe, all other keys are hashed totally randomly. We consider unsuccessful searches, i.e. the search key q is not stored in the hash table. The query time for q is the number of cells considered from $h(q)$ up to the first empty cell. If, for some d, the interval $Q = (h(q) - d, h(q)]$ has $2d$ keys, then the search time is $\Omega(d)$.

Let $d = 2\sqrt{n}$; this is a power of two dividing b. In our construction, we first pick the hash $h(q)$ uniformly. We then divide the range into $\sqrt{n}$ intervals of length d, of the form $(h(q) + i \cdot d, h(q) + (i+1)d]$, wrapping around modulo b. One of these intervals is exactly Q.

We prescribe the distribution of keys between the intervals; the distribution within each interval will be fully random. To place $2d = 4\sqrt{n}$ keys in the query interval with constant probability, we mix among two strategies with constant probabilities (to be determined):

S_1: Spread keys evenly, with $\sqrt{n}$ keys in each interval.

S_2: Pick 4 intervals including the query interval, and gather all $4\sqrt{n}$ keys in a random one of these. All other intervals get $\sqrt{n}$ keys. With probability 1/4, it is the query interval that gets overfilled, and the search time is $\Omega(\sqrt{n})$.

To prove that the hash function is 2-independent, we need to consider pairs of two stored keys, and pairs involving the query and one stored key. In either case, we can just look at the distribution into intervals, since the position within an interval is truly random. Furthermore, we only need to understand the probability of the two keys landing in the same interval (which we call a "collision"). By

the above process, if two keys do not collide, they will actually be in uniformly random distinct intervals.

Since store keys are symmetric, the probability of q and x colliding is given by the expected number of items in Q, which is exactly $d/2$. Thus $h(q)$ and $h(x)$ are independent. To analyze pairs of the form (x, y), we will compute the expected number of collisions among stored keys. This will turn out to be $\binom{n}{2}/\sqrt{n}$, proving that x and y collide with probability $1/\sqrt{n}$, and thus, are independent.

In strategy S_1, we get the smallest possible number of collisions: $\sqrt{n}\binom{\sqrt{n}}{2} = \frac{1}{2}n^{1.5} - \frac{1}{2}n$. Compared to $\binom{n}{2}/\sqrt{n} = \frac{1}{2}n^{1.5} - \frac{1}{2}\sqrt{n}$, this is too few by almost $n/2$. In S_2, we get $(\sqrt{n}-4)\binom{\sqrt{n}}{2} + \binom{4\sqrt{n}}{2} = \frac{1}{2}n^{1.5} + \frac{11}{2}n$, which is too large by almost $5.5n$. To get the right expected number of collisions, we use S_2 with probability $\frac{5.5n+o(n)}{(0.5+5.5)n+o(n)} = \frac{11}{12} \pm o(1)$.

It is not hard to prove an upper bound of $O(\sqrt{n})$ on the expected query cost: a higher query time implies too many collisions for 2-independence. Formally, divide into $\sqrt{n}$ intervals of length $d = 2\sqrt{n}$. If the query cost is td, it must pass at least $(t-2)d$ keys in intervals filled with at least d keys. Also, if we have k keys in such full intervals, the number of collisions is $\Omega(k\sqrt{n})$ above the minimum possible. Thus, if the expected query cost is $\omega(d)$, the the expected number of extra collisions is $\omega(n)$, contradicting 2-universality.

3 Construction Time $\Omega(n \lg n)$ with 3-Independence

We will now construct a 3-independent family of hash functions, such that the time to insert n items into a hash table is $\Omega(n \lg n)$. As before, we assume the array size b is a power of two, and set $n = \lceil \frac{2}{3}b \rceil$. Since we are looking at the total cost of n insertions, if some interval of length d gets $d+s$ keys (overflow of s), then these $d+s$ insertions cost $\Omega(s^2)$. We will add up such squared overflow costs over disjoint intervals, and demonstrate a total cost of $\Omega(n \lg n)$.

We imagine a perfect binary tree spanning the array. Our hash function will recursively distribute keys from a node to its two children, starting at the root. Nodes run independent random distribution processes. Then, if each node makes a k-independent distribution, overall the function is k-independent.

For a node, we mix between two strategies:

S_1**:** Distribute the keys evenly between the two children, breaking ties randomly.
S_2**:** Give all the keys to one of the children, chosen randomly.

Our first goal is to determine the correct probability for the second strategy, p^{S2}, such that the distribution process is 3-independent. Then we will calculate the cost it induces on linear probing.

3.1 Characterizing k-Independence

Our randomized procedure treats keys symmetrically, and ignores the distinction between left/right children. We call such distributions *fully symmetric*. Say the current node has to distribute $2m$ keys to its two children (m need not be

integral). Let X_a be the indicator random variable for key a ending in the left child, and $X = \sum_a X_a$. By symmetry of the children, $\mathbf{E}[X_a] = \frac{1}{2}$, so $\mathbf{E}[X] = m$. The k^{th} moment is $F_k = \mathbf{E}[(X-m)^k]$. Also define $p_k = \Pr[X_1 = \cdots = X_k = 1]$ (by symmetry, any k distinct keys yield the same value).

Lemma 1. *A fully symmetric distribution is k-independent iff $p_i = 2^{-i}$ for all $i = 2, \ldots, k$.*

Proof. We need to show that, for any $(x_1, \ldots, x_k) \in \{0,1\}^k$, $\Pr[(X_1 = x_1) \wedge \cdots \wedge (X_k = x_k)] = 2^{-k}$. By symmetry of the keys, we can sort the vector to $x_1 = \cdots = x_t = 1$ and $x_{t+1} = \cdots = x_k = 0$. Let $P_{k,t}$ be the probability that such a vector is seen.

We use induction on k. In the base case, $P_{1,0} = P_{1,1} = \frac{1}{2}$ by symmetry. For $k \geq 2$, we start with $P_{k,k} = p_k = 2^{-k}$. We then use induction for $t = k-1$ down to $t = 0$. The induction step is simply: $P_{k,t} = P_{k-1,t} - P_{k,t+1} = 2^{-(k-1)} - 2^{-k} = 2^{-k}$. Indeed, $\Pr[X_{1..t} = 1 \wedge X_{t+1..k} = 0]$ can be computed as the difference between $\Pr[X_{1..t} = 1 \wedge X_{t+1..k-1} = 0]$ (measured by $P_{k-1,t}$) and $\Pr[X_{1..t} = 1 \wedge X_{t+1..k-1} = 0 \wedge X_k = 1]$ (measured by $P_{k,t+1}$).

Based on this lemma, we can also give a characterization based on moments. First observe that any odd moment is necessarily zero, as $\Pr[X = m + \delta] = \Pr[X = m - \delta]$ by symmetry of the children.

Lemma 2. *A fully symmetric distribution is k-independent iff its even moments between F_2 and F_k coincide with the moments of the truly random distribution.*

Proof. We will show that $p_2, \ldots, p_k$ are determined by $F_2, \ldots, F_k$. Thus, any distribution that has the same moments as a truly random distribution, will have the same values $p_2, \ldots, p_k$ as the truly random distribution ($p_i = 2^{-i}$ as in Lemma 1).

Let $n^{\bar{k}} = n(n-1)\ldots(n-k+1)$ be the falling factorial. We use induction on k with the following formula:

$$p_k = F_k/(2m)^{\bar{k}} \;+\; f_k(m, F_2, .., F_{k-1}), \qquad \text{for some function } f_k.$$

To see this, note that $F_k = \mathbf{E}[(X-m)^k] = \mathbf{E}[X^k] + f\big(m, \mathbf{E}[X^2], \ldots, \mathbf{E}[X^{k-1}]\big)$. But $\mathbf{E}[X^k] = (2m)^{\bar{k}} p_k + f_1(m,k)p_{k-1} + f_2(m,k)p_{k-2} + \ldots$. Here, the factors $f_i(m,k)$ count the number of ways to select k out of $2m$ keys, with i duplicates. □

3.2 The Cost of Linear Probing

By the above characterization, a mix of S_1 and S_2 is 3-independent iff it has the correct 2^{nd} moment $F_2 = \frac{m}{2}$. In strategy S_1, $X = m \pm 1$ (due to rounding errors if $2m$ is odd), so $F_2^{S1} \leq 1$. In S_2 (all to one child), $|X - m| = m$ so $F_2^{S2} = m^2$. Hence, the proper balancing is $p^{S2} = \frac{2}{m} \pm O(\frac{1}{m^2})$, yielding a 2^{nd} moment of $m/2$.

We now calculate the cost in terms of squared overflows. As long as the recursive steps spread the keys evenly, the load stays around 2/3. However, a first time

we collect all keys into one child, that interval of the array will get a load of $4/3$. This is an overflow of $\Omega(m)$ keys, thus a cost of $\Omega(m^2)$. Since $p^{S2} = \Theta(1/m)$, the expected cost induced by the node is $\Theta(m)$.

However, to avoid double charging, we may only consider the node if there has been not collection in one of his ancestors. As long as S_1 applies, the number of keys at depth i is $2m \approx n/2^i$, so the probability of the collection strategy is $p^{S2} = \Theta(1/m) = \Theta(2^i/n)$. The probability that a node at depth i is still relevant (no collection among his ancestors) is at least $1 - \sum_{j=0}^{i-1} \Theta(2^j/n) = 1 - \Theta(2^i/n) \geq \frac{1}{2}$ for $i \ll \lg n$. We conclude that the expected cost of each node is linear in the size of its subtree. Hence, the total expected cost is $\Omega(n \lg n)$.

4 Expected Query Time $\Omega(\lg n)$ with 4-Independence

Due to space limitations, the proof appears only in the full version of the paper. The proof is a combination of the ideas for 2-independence and 3-independence, plus the (severe) complications that arise. As for 2-independence, we will first choose $h(q)$ and then make the stored keys cluster preferentially around $h(q)$. This clustering will actually be the 3-independent type of clustering from before, but done only (mostly) on the path from the root to $h(q)$. Since this is used for so few nodes, we can balance the 4^{th} moment to give 4-independence, by doing a slightly skewed distribution in the nodes off the query path.

5 Minwise Independence via k-Independence

We will show that it is limited how good minwise independence we can achieve based on k-independent hashing. For a given k, our goal is to construct a k-independent distribution over n regular keys and a query key q, such that the probability that q gets the minimal hash value is $\frac{1}{n+1}\big(1 + 2^{-O(k)}\big)$.

We assume that k is even and divides n. Each hash value will be uniformly distributed in the unit interval $[0, 1)$. Discretizing this continuous interval does not affect any of the calculations below, as long as precision $2 \lg n$ or more is used (making the probability of a non-unique minimum vanishingly small).

For our construction, we divide the unit interval into $\frac{n}{k}$ subintervals of the form $\big[i\frac{k}{n}, (i+1)\frac{k}{n}\big)$. The regular keys are distributed totally randomly between these subintervals. Each subinterval I gets k regular keys in expectation. We say that I is *exact* if it gets exactly k regular keys. Whenenver I is not exact, the regular keys are placed totally randomly within it.

The distribution inside an exact interval I is dictated by a parity parameter $P \in \{0, 1\}$. We break I into two equal halves, and distribute the k keys into these halves randomly, conditioned on the parity in the first half being P. Within its half, each key gets an independent random value. If P is fixed, this process is $k - 1$ independent. Indeed, one can always deduce the half of a key x based on knowledge of $k - 1$ keys, but the location of x is totally uniform if we only know about $k - 2$ keys. If the parity parameter P is uniform in $\{0, 1\}$ (but possibly dependent among exact intervals), the overall distribution is still k-independent.

The query is generated independently and uniformly. For each exact interval I, if the query is inside it, we set its parity parameter $P_I = 0$. If I is exact but the query is outside it, we toss a biased coin to determine the parity, with $\Pr[P_I = 0] = (\frac{1}{2} - \frac{k}{n})/(1 - \frac{k}{n})$. Any fixed exact interval receives the query with probability $\frac{k}{n}$, so overall the distribution of P_I is uniform.

We claim that the overall process is k-independent. Uniformity of P_I implies that the distribution of regular keys is k-independent. In the case of q and $k-1$ regular keys, we also have full independence, since the distribution in an interval is $(k-1)$-independent even conditioned on P.

It remains to calculate the probability of q being the minimum under this distribution. First we assume that the query landed in an exact interval I, and calculate $p_{\min}$, the probability that q takes the minimum value within I. Define the random variable X as the number of regular keys in the first half. By our process, X is always even.

If $X = x > 0$, q is the minimum only if it lands in the first half (probability $\frac{1}{2}$) and is smaller than the x keys already there (probability $\frac{1}{x+1}$). If $X = 0$, q is the minimum either if it lands in the first half (probability $\frac{1}{2}$), or if it lands in the second half, but is smaller than everybody there (probability $\frac{1}{2(k+1)}$). Thus,

$$p_{\min} = \Pr[X = 0] \cdot \left(\tfrac{1}{2} + \tfrac{1}{2(k+1)}\right) + \sum_{x=2,4,..,k} \Pr[X = x] \cdot \tfrac{1}{2(x+1)}$$

To compute $\Pr[X = x]$, we can think of the distribution into halves as a two step process: first $k-1$ keys are distributed randomly; then, the last key is placed to make the parity of the first half even. Thus, $X = x$ if either x or $x-1$ of the first $k-1$ keys landed in the first half. In other words:

$$\Pr[X = x] = \tbinom{k-1}{x}/2^{k-1} + \tbinom{k-1}{x-1}/2^{k-1} = \tbinom{k}{x}/2^{k-1}$$

No keys are placed in the first half iff none of the first $k-1$ keys land there; thus $\Pr[X = 0] = 1/2^{k-1}$. We obtain:

$$p_{\min} = \frac{1}{2^k(k+1)} + \frac{1}{2^k} \sum_{x=0,2,..,k} \frac{1}{x+1} \binom{k}{x}$$

But $\frac{1}{x+1}\binom{k}{x} = \frac{1}{k+1}\binom{k+1}{x+1}$. Since $k+1$ is odd, the sum over all odd binomial coefficients is exactly $2^{k+1}/2$ (it is equal to the sum over even binomial coefficients, and half the total). Thus, $p_{\min} = \frac{1}{2^k(k+1)} + \frac{1}{k+1}$, i.e. q is the minimum with a probability that is too large by a factor of $1 + 2^{-k}$.

We are now almost done. For q to be the minimum of all keys, it has to be in the minimum non-empty interval. If this interval is exact, our distribution increases the chance that q is minimum by a factor $1 + 2^{-k}$; otherwise, our distribution is completely random in the interval, so q is minimum with its fair probability. Let Z be the number of regular keys in q's interval, and let $\mathcal{E}$ be the event that q's interval is the minimum non-empty interval. If the distribution

were truly random, then q would be minimum with probability:

$$\frac{1}{n+1} = \sum_z \Pr[Z = z] \cdot \Pr[\mathcal{E} \mid Z = z] \cdot \frac{1}{z+1}$$

In our tweaked distribution, q is minimum with probability:

$$\sum_{z \neq k} \Pr[Z = z] \cdot \Pr[\mathcal{E} \mid Z = z] \cdot \frac{1}{z+1} + \Pr[Z = k] \cdot \Pr[\mathcal{E} \mid Z = k] \cdot \frac{1 + 2^{-k}}{k+1}$$
$$= \frac{1}{n+1} \; + \; \Pr[Z = k] \cdot \Pr[\mathcal{E} \mid Z = k] \cdot \frac{2^{-k}}{k+1}$$

But Z is a binomial distribution with n trials and mean k; thus $\Pr[Z = k] = \Omega(1/\sqrt{k})$. Furthermore, $\Pr[\mathcal{E} \mid Z = k] \geq \frac{k}{n}$, since q's interval is the very first with probability $\frac{k}{n}$ (and there is also a nonzero chance that it is not the first, but all interval before are empty). Thus, the probability is off by an additive term $\frac{\Omega(2^{-k}/\sqrt{k})}{n}$. This translates into a multiplicate factor of $1 + 2^{-O(k)}$.

References

1. Alon, N., Nussboim, A.: k-wise independent random graphs. In: Proc. 49th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 813–822 (2008)
2. Black, J.R., Martel, C.U., Qi, H.: Graph and hashing algorithms for modern architectures: Design and performance. In: Proc. 2nd International Workshop on Algorithm Engineering (WAE), pp. 37–48 (1998)
3. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations. Journal of Computer and System Sciences 60(3), 630–659 (2000); See also STOC 1998
4. Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G.: Syntactic clustering of the web. Computer Networks 29, 1157–1166 (1997)
5. Cohen, E.: Size-estimation framework with applications to transitive closure and reachability. Journal of Computer and System Sciences 55(3), 441–453 (1997); See also STOC 1994
6. Dietzfelbinger, M.: Universal hashing and k-wise independent random variables via integer arithmetic without primes. In: Proc. 13th Symposium on Theoretical Aspects of Computer Science (STACS), pp. 569–580 (1996)
7. Heileman, G.L., Luo, W.: How caching affects hashing. In: Proc. 7th Workshop on Algorithm Engineering and Experiments (ALENEX), pp. 141–154 (2005)
8. Indyk, P.: A small approximately min-wise independent family of hash functions. Journal of Algorithms 38(1), 84–90 (2001); See also SODA 1999
9. Knuth, D.E.: Notes on open addressing (1963) (Unpublished memorandum), http://citeseer.ist.psu.edu/knuth63notes.html
10. Pagh, A., Pagh, R., Ružić, M.: Linear probing with constant independence. SIAM Journal on Computing 39(3), 1107–1120 (2009); See also STOC 2007
11. Pagh, R., Rodler, F.F.: Cuckoo hashing. Journal of Algorithms 51(2), 122–144 (2004); See also ESA 2001
12. Pătraşcu, M., Thorup, M.: The power of simple tabulation-based hashing (2010) (manuscript)

13. Schmidt, J.P., Siegel, A.: The analysis of closed hashing under limited randomness. In: Proc. 22nd ACM Symposium on Theory of Computing (STOC), pp. 224–234 (1990)
14. Schmidt, J.P., Siegel, A., Srinivasan, A.: Chernoff-Hoeffding bounds for applications with limited independence. SIAM Journal on Discrete Mathematics 8(2), 223–250 (1995); See also SODA 1993
15. Siegel, A., Schmidt, J.P.: Closed hashing is computable and optimally randomizable with universal hash functions. Technical Report TR1995-687, Currant Institute (1995)
16. Thorup, M.: Even strongly universal hashing is pretty fast. In: Proc. 11th ACM/SIAM Symposium on Discrete Algorithms (SODA), pp. 496–497 (2000)
17. Thorup, M., Zhang, Y.: Tabulation based 4-universal hashing with applications to second moment estimation. In: Proc. 15th ACM/SIAM Symposium on Discrete Algorithms (SODA), pp. 615–624 (2004)
18. Thorup, M., Zhang, Y.: Tabulation based 5-universal hashing and linear probing. In: Proc. 12th Workshop on Algorithm Engineering and Experiments, ALENEX (2009)
19. Wegman, M.N., Carter, L.: New classes and applications of hash functions. Journal of Computer and System Sciences 22(3), 265–279 (1981); see also FOCS 1979

Covering and Packing in Linear Space*

Andreas Björklund[1], Thore Husfeldt[1,2], Petteri Kaski[3], and Mikko Koivisto[4]

[1] Lund University, Department of Computer Science,
P.O.Box 118, SE-22100 Lund, Sweden
andreas.bjorklund@yahoo.se
[2] IT University of Copenhagen,
2300 Copenhagen S, Denmark
thore@itu.dk
[3] Helsinki Institute for Information Technology HIIT,
Aalto University, School of Science and Technology,
Department of Information and Computer Science
PO Box 15400, FI-00076 Aalto, Finland
petteri.kaski@tkk.fi
[4] Helsinki Institute for Information Technology HIIT,
Department of Computer Science, University of Helsinki,
P.O.Box 68, FI-00014 University of Helsinki, Finland
mikko.koivisto@cs.helsinki.fi

Abstract. Given a family of subsets of an n-element universe, the k-cover problem asks whether there are k sets in the family whose union contains the universe; in the k-packing problem the sets are required to be pairwise disjoint and their union contained in the universe. When the size of the family is exponential in n, the fastest known algorithms for these problems use inclusion–exclusion and fast zeta transform, taking time and space 2^n, up to a factor polynomial in n. Can one improve these bounds to only linear in the size of the family? Here, we answer the question in the affirmative regarding the space requirement, while not increasing the time requirement. Our key contribution is a new fast zeta transform that adapts its space usage to the support of the function to be transformed. Thus, for instance, the chromatic or domatic number of an n-vertex graph can be found in time within a polynomial factor of 2^n and space proportional to the number of maximal independent sets, $O(1.442^n)$, or minimal dominating sets, $O(1.716^n)$, respectively. Moreover, by exploiting some properties of independent sets, we reduce the space requirement for computing the chromatic polynomial to $O(1.292^n)$. Our algorithms also parallelize efficiently.

1 Introduction

Brute-force algorithms typically use lots of time but only little space. For instance, a straightforward algorithm for the traveling salesman problem (TSP)

* This research was supported in part by the Swedish Research Council, project "Exact Algorithms" (A.B., T.H.), and the Academy of Finland, Grants 117499 (P.K.) and 125637 (M.K.).

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 727–737, 2010.

visits every possible permutation of the n cities, requiring about $n!$ computational steps and a storage for about n cities and two real numbers (in addition to the input). Designing faster algorithms is sometimes possible by trading space against time. Indeed, this is precisely what Bellman's [1,2] and Held and Karp's [7] dynamic programming treatment of TSP does by tabulating partial solutions across all subsets of the n cities: both the runtime and the space requirement grow as 2^n. A similar story can be told about coloring n-vertex graphs, or set covering more generally, albeit the 2^n bounds were discovered only quite recently [3]. Reducing the space requirement for these problems appears challenging, and has been so far achieved only at the cost of increasing the running time [5,9].

In this paper, we provide an input-sensitive characterization of the space–time tradeoff for the set cover problem. Regarding the time requirement, the best one can hope for is an upper bound linear in the size of the input, for all input must be read in the worst case. While no such lower bound is obvious for the space requirement, one may, again, regard a linear upper bound as a plausible goal. Since a set family over an n-set universe may contain an order of 2^n members, the known upper bounds are optimal in the worst case. For the current techniques, however, the $O^*(2^n)$ time and space bounds are tight even if the given set family is much smaller; throughout the paper the O^* notation hides a factor polynomial in n. Here, we show that the set cover problem can be solved in time $O^*(2^n)$ using space only about linear in the size of the set family. Applications to graph coloring and domatic partitioning yield space bounds of the form $O(C^n)$ with $C < 2$, as outlined in the sequel.

When making these claims we assume the general case where the set family is given explicitly in the input. It goes without saying that in many concrete problems, such as graph coloring or domatic partitioning, the set family is represented implicitly, for example in terms of a graph.

Our study actually concerns the counting variant of the set cover problem. A *k-cover* over a family $\mathcal{F}$ of subsets of an n-element universe U is a tuple of k members of $\mathcal{F}$ whose union contains U. Given $\mathcal{F}$ and k, the counting problem asks the number of k-covers over $\mathcal{F}$, denoted by $c_k(\mathcal{F})$. We start with the inclusion–exclusion formula [3],

$$c_k(\mathcal{F}) = \sum_{X \subseteq U} (-1)^{|U \setminus X|} a(X)^k \,, \tag{1}$$

where $a(X)$ is the number of subsets $Y \subseteq X$ that belong to $\mathcal{F}$. The key observation is that given $\mathcal{F}$, the numbers $a(X)$, for all $X \subseteq U$, can be listed (in some order) in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$.

It is useful to formulate this result slightly more generally in terms of the zeta transform on the subset lattice, as follows. If f is a function from the subsets of U to a ring R, then the zeta transform of f, denoted as $f\zeta$, is defined by

$$f\zeta(X) = \sum_{Y \subseteq X} f(Y), \qquad X \subseteq U \,.$$

See Fig. 1 for an illustration of the zeta transform.

Theorem 1. *Suppose f vanishes outside $\mathcal{F}$ and the members of $\mathcal{F}$ can be listed in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$. Then the values $f\zeta(X)$, for $X \subseteq U$, can be listed in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$.*

This result, which we will prove in Sect. 3, allows us to easily extend the result for set covers to an analogous result for set partitions and set packings; a *k-partition* (*k-packing*) over $\mathcal{F}$ is a tuple of k pairwise disjoint member of $\mathcal{F}$ whose union equals (is contained in) the universe U. Thus, given Theorem 1, we have the following.

Theorem 2. *Let $\mathcal{F}$ be a family of subsets of an n-element universe U, and let k be an integer. Suppose $\mathcal{F}$ can be listed in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$. Then the k-covers, k-packings, and k-partitions over $\mathcal{F}$ can be counted in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$.*

We illustrate this result with some immediate implications to graph coloring and domatic partitioning. The *chromatic number* of a graph is the smallest integer k such that there exists a proper k-coloring of the graph, that is, a mapping σ from the vertices of the graph to $\{1, 2, \ldots, k\}$ such that $\sigma(u) \neq \sigma(v)$ if u and v are adjacent in the graph. To test if the chromatic number is k or smaller, it clearly suffices to count the covers of the vertices by k independent sets of the graph; a subset of vertices is an independent set if it does not contain two adjacent vertices. In fact, it suffices to count the covers of the vertices by k *maximal* independent sets; an independent set is maximal if it is not a subset of any other independent set. The maximal independent sets can be trivially listed in time $O^*(2^n)$ and space linear in their number. Because a graph with n vertices can have at most $3^{n/3} \approx 1.44225^n$ maximal independent sets [10], Theorem 2 gives us the following.

Corollary 1. *The chromatic number of a given n-vertex graph can be found in time $O^*(2^n)$ and space $O(1.443^n)$.*

Analogous reasoning applies to domatic partitioning. The *domatic number* of a graph is the largest integer k such that the vertices of the graph be partitioned into k pairwise disjoint dominating sets; a subset of vertices D is a dominating set if every vertex not in D is adjacent to at least one vertex in D. To test if the domatic number is k or larger, it suffices to count the k-packings over the dominating sets of the graph. Again, it actually suffices to count the k-packings over the *minimal* dominating sets; a dominating set is minimal if it contains no other dominating set. Because a graph with n vertices can have at most 1.716^n minimal dominating sets, which can be listed in time $O(1.716^n)$ [6], we have the following.

Corollary 2. *The domatic number of a given n-vertex graph can be found in time $O^*(2^n)$ and space $O(1.716^n)$.*

It should be noted that the computation of the chromatic or domatic number are, in essence, decision problems, even though the counting approach is crucial for

obtaining the reduced space requirement. If Theorem 2 is applied, for example, to the computation of the *chromatic polynomial*, which at k evaluates to the number of proper k-colorings, then the space bound $O^*(2^n)$ is tight, since it does not suffice to consider only maximal independent sets. In this light, we find it somewhat surprising that the chromatic polynomial can, however, be computed in much less space by using a variant of the linear-space zeta transform that exploits the special structure of independent sets. Indeed, in Sect. 5 we prove the following bound, which improves upon Corollary 1.

Theorem 3. *The chromatic polynomial of a given n-vertex graph can be found in time $O^*(2^n)$ and space $O(1.292^n)$.*

However, the algorithm behind Corollary 1 remains interesting for finding the chromatic number, as most graphs have a lot fewer maximal independent sets than the Moon–Moser bound $3^{n/3}$ predicts.

Apart from the space savings, our algorithms have also another feature that should be relevant for practical implementations: they admit efficient parallelization. It will be immediate from the descriptions in Sects. 3 and 4 that the algorithms can be executed in parallel on $O^*(2^n/S)$ processors, each using time and space $O^*(S)$, where S varies as given in the space bounds of Theorems 1 and 2 and Corollaries 1 and 2; for the chromatic polynomial some extra space is needed compared to Theorem 3: $O^*(2^{n/2}) = O(1.415^n)$ processors, each using time and space $O^*(2^{n/2})$. This capability for parallel computation is in sharp contrast to the previous algorithms [3], for which efficient parallelization to exponentially many processors seems not possible.

Remark. In the statements above and their proofs in the remainder sections, we ignore the polynomial factors for simplicity. We note, however, that the hidden factors are relatively small. For instance, in Theorem 1 the actual storage requirement is $O(|\mathcal{F}|n)$ bits and ring elements, assuming each member of $\mathcal{F}$ is represented naturally by n bits. Likewise, in Theorem 2 $O(|\mathcal{F}|n)$ bits suffice if we resort to space-efficient manipulation of the involved polynomials, that is, evaluation–interpolation, each evaluation modulo small relative primes and using the Chinese Remainder Theorem. We omit a more detailed consideration of these standard techniques in this paper.

2 Preliminaries

We adopt the following conventions. Let U be a universe of n elements. Denote by 2^U the set of all subsets of U. Let R be an algebraic ring. Let $f : 2^U \to R$ be a function. The *(down-)zeta transform* of f is the function $f\zeta : 2^U \to R$, defined for all $X \subseteq U$ by $f\zeta(X) = \sum_{Y \subseteq X} f(Y)$. The *up-zeta transform* of f is the function $f\zeta' : 2^U \to R$, defined for all $X \subseteq U$ by $f\zeta'(X) = \sum_{X \subseteq Y} f(Y)$.

We employ Iverson's bracket notation, that is, for a logical proposition P, we write $[P]$ to indicate a 1 if P is true and a 0 if P is false.

We recall that there is an algorithm, *the fast zeta transform* [8,4], that computes the function $f\zeta$ from the function f in time and space $O^*(2^n)$, where we assume that the arithmetic operations in the ring R take time $O^*(1)$ and each ring element takes $O^*(1)$ space.

The Fast Zeta Transform. Let the universe be $U = \{1, 2, \ldots, n\}$ and let $f : 2^U \to R$ be given as input. The algorithm pseudocode is as follows:

1. Set $f_0 \leftarrow f$.
2. For each $j = 1, 2, \ldots, n$ do:
   (a) For each $X \subseteq U$, set $f_j(X) \leftarrow [j \in X] f_{j-1}(X \setminus \{j\}) + f_{j-1}(X)$.
3. Give the output $f\zeta \leftarrow f_n$.

An analogous algorithm is easy to derive for the up-zeta transform.

3 The Zeta Transform in Linear Space and $O^*(2^n)$ Time

This section proves Theorem 1.

Let us fix a bipartition of the universe U,

$$U = U_1 \cup U_2, \qquad U_1 \cap U_2 = \emptyset, \qquad |U_1| = n_1, \qquad |U_2| = n_2\,, \tag{2}$$

for integers n_1 and n_2 yet to be fixed. We now execute the following algorithm; see Fig. 2.

The Linear-Space Fast Zeta Transform. Let a family $\mathcal{F} \subseteq 2^U$ and a function $f : 2^U \to R$ that vanishes outside $\mathcal{F}$ be given as input. We execute the following algorithm to output $f\zeta(X)$ for each $X \subseteq U$, with comments delimited by double braces "{{" and "}}":

1. For each $X_1 \subseteq U_1$ do:
   (a) For each $Y_2 \subseteq U_2$, set $g(Y_2) \leftarrow 0$.
   {{ This step takes $O^*(2^{n_2})$ time and space. }}
   (b) For each $Y \in \mathcal{F}$, if $Y \cap U_1 \subseteq X_1$ then set $g(Y \cap U_2) \leftarrow g(Y \cap U_2) + f(Y)$.
   {{ This step takes $O^*(|\mathcal{F}|)$ time and $O^*(2^{n_2})$ space. }}
   (c) Compute $h \leftarrow g\zeta$ using the fast zeta transform on 2^{U_2}.
   {{ This step takes $O^*(2^{n_2})$ time and space. }}
   (d) For each $X_2 \subseteq U_2$, output the value $h(X_2)$ as the value $f\zeta(X_1 \cup X_2)$.

Note that the algorithm evaluates f only at $\mathcal{F}$. Moreover, we can iterate over the elements of $\mathcal{F}$ in arbitrary order.

We establish the correctness of the algorithm by analyzing the contents of the array h during each iteration of the loop over $X_1 \subseteq U_1$:

Lemma 1. *For each fixed $X_1 \subseteq U_1$, we have $h(X_2) = f\zeta(X_1 \cup X_2)$ for all $X_2 \subseteq U_2$.*

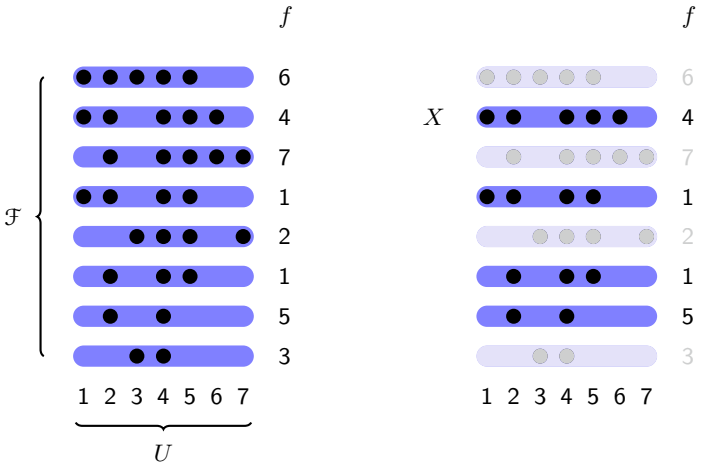


Fig. 1. Left: a set family $\mathcal{F}$ over $U = \{1, 2, \ldots, 7\}$, and values $f(Y) \neq 0$ for every $Y \in \mathcal{F}$. Right: For $X = \{1, 2, 4, 5, 6\}$, the value $f\zeta(X)$ is the sum over its subsets, $4 + 1 + 1 + 5 = 11$.

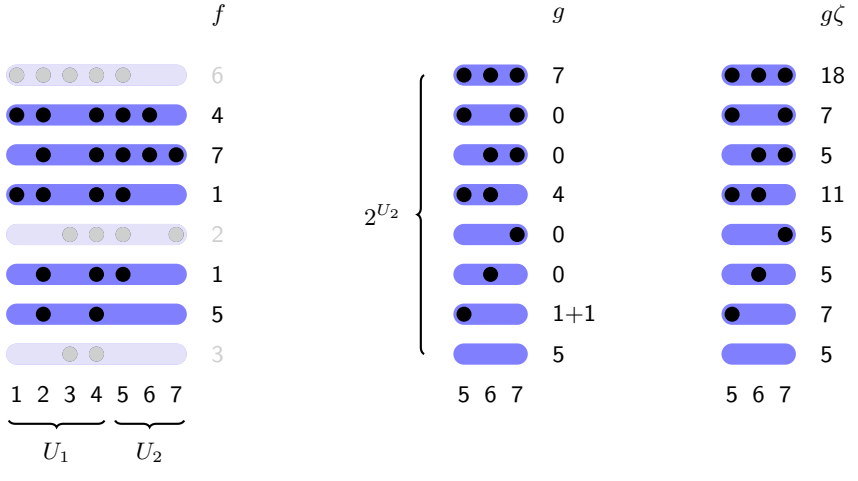


Fig. 2. The linear space zeta transform. U is partitioned into U_1 and U_2 with $|U_2| = \log |\mathcal{F}|$. We iterate over all $X_1 \subseteq U_1$; here we show $X_1 = \{1, 2, 4\}$. Left: We consider only the sets $Y \in \mathcal{F}$ with $Y \cap U_1 \subseteq X_1$. Middle: For each of these sets, add its value $f(Y)$ to $g(Y \cap U_2)$. Right: Compute the zeta transform $g\zeta$ on U_2. The result contains the values of $f\zeta$ for all X for which $X \cap U_1 = X_1$. For example, $f\zeta(\{1, 2, 4, 5, 6\}) = 11$. The central point of the analysis is that the most space-expensive operation, the exponential-space fast zeta transform, operates only on U_2, so the whole algorithm runs in space $2^{|U_2|} = |\mathcal{F}|$.

Proof. Expanding the assignments in the algorithm, we have

$$\begin{aligned} h(X_2) &= \sum_{Y_2 \subseteq X_2} g(Y_2) \\ &= \sum_{Y_2 \subseteq X_2} \sum_{Y \in \mathcal{F}} [Y \cap U_1 \subseteq X_1][Y \cap U_2 = Y_2] f(Y) \\ &= \sum_{Y \in \mathcal{F}} [Y \cap U_1 \subseteq X_1][Y \cap U_2 \subseteq X_2] f(Y) \\ &= \sum_{\substack{Y \in \mathcal{F} \\ Y \subseteq X_1 \cup X_2}} f(Y) \\ &= \sum_{Y \subseteq X_1 \cup X_2} f(Y) \\ &= f\zeta(X_1 \cup X_2)\,. \end{aligned}$$

□

Observe that the algorithm runs in $O^*(2^{n_1}(2^{n_2} + |\mathcal{F}|))$ time and $O^*(2^{n_2})$ space. We now fix the values n_1 and n_2 to $n_2 = \lceil \log_2 |\mathcal{F}| \rceil$ and $n_1 = n - n_2$. The algorithm thus runs in $O^*(2^n)$ time and $O^*(|\mathcal{F}|)$ space. Note also that because the computations are independent for each $X_1 \subseteq U_1$, they can be executed in parallel on $O^*(2^n/|\mathcal{F}|)$ processors.

4 Coverings, Packings, and Partitions

This section proves Theorem 2.

To compute the number of k-coverings we need to evaluate (1). Note that a equals $f\zeta$, where f is $\mathcal{F}$'s characteristic function, $f(Y) = [Y \in \mathcal{F}]$. By Theorem 1 we can list all values $a(X)$ for $X \subseteq U$ in some order within the desired time and space bounds; while doing so we accumulate their kth powers with the sign given by (1).

We turn to counting the k-partitions. The idea is to modify the function a above so that it controls the size of the counted members of the set family [3]. This can be handily formulated [4] by replacing $a(X)$ by a polynomial over an indeterminate z:

$$d_k(\mathcal{F}) = \sum_{X \subseteq U} (-1)^{|U \setminus X|} \Big(a_0(X) + a_1(X) z + \cdots + a_n(X) z^n \Big)^k ,$$

where the coefficient $a_j(X)$ is the number of subsets $Y \subseteq X$ that belong to $\mathcal{F}$ and are of size j. Now $d_k(\mathcal{F})$ is a polynomial whose coefficient of the monomial z^n is the number of k-partitions [4]. To evaluate this expression, we note that the polynomial $a(X)$ equals $g\zeta(X)$, where $g(Y) = [Y \in \mathcal{F}] z^{|Y|}$. Now, the linear-space fast zeta transform (Theorem 1, Sect. 3) operates in a ring of polynomials and lists the polynomials $a(X)$ for all $X \subseteq U$ in the desired time and space.[1]

[1] To save some polynomial factors in both time and space, the arithmetic should actually not be carried out directly with polynomials. Instead, the polynomials can be evaluated at sufficiently many small integers, and finally the coefficients of $d_k(\mathcal{F})$ are recovered via interpolation and the Chinese Remainder Theorem.

Finally, the number of k-packings can be viewed as the number of $(k+1)$-partitions with k members from $\mathcal{F}$, the $(k+1)$th member being an arbirary subset of U. The expression corresponding to (1) becomes

$$p_k(\mathcal{F}) = \sum_{X \subseteq U} (-1)^{|U \setminus X|} (1+z)^{|X|} \left(\sum_{j=0}^{n} a_j(X) z^j \right)^k ,$$

and the coefficient of z^n gives the number of k-packings.

5 The Chromatic Polynomial in Time $O^*(2^n)$ and Space $O(1.2916^n)$

This section proves Theorem 3.

Let G be a graph with vertex set V, $|V| = n$. For a positive integer k, denote by $\chi_G(k)$ the number of proper k-colorings of the vertices of G, that is, the number of mappings σ from V to $\{1, 2, \ldots, k\}$ so that $\sigma(u) \neq \sigma(v)$ holds whenever u and v are adjacent in G. It is well known that the integer function $\chi_G(t)$ admits representation as a polynomial of degree n with integer coefficients. In particular, $\chi_G(t)$ is called the *chromatic polynomial* of G.

To compute $\chi_G(t)$ from a given G, it suffices to evaluate $\chi_G(k)$ for $n+1$ distinct positive integers k and then recover the polynomial in t by interpolation.

Thus, our task reduces to counting the number of proper k-coloring of G. Equivalently, our task is to count the number of ordered partitions (with empty parts permitted) of the vertices of G into k independent sets. Put otherwise, it suffices to compute for every $X \subseteq V$ the z-polynomial

$$i(X) = \sum_{Y \subseteq X} z^{|Y|} [Y \text{ is independent in } G] ,$$

which enables us to recover the number of k-colorings of G as the coefficient of z^n in the polynomial

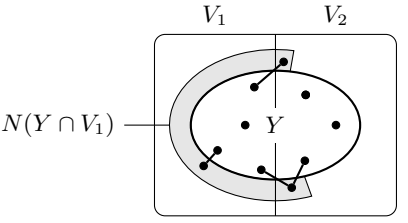
$$r = \sum_{X \subseteq V} (-1)^{n-|X|} i(X)^k .$$

Our improved space requirement stems from an algorithm that evaluates the polynomials $i(X)$ in two parts. To this end, partition the vertex set V into two disjoint sets, V_1 and V_2, with $|V_1| = n_1$ and $|V_2| = n_2$. Let $n_1 = \lceil n(\log 2)/(\log 3) \rceil$ and $n_2 = n - n_1$. Observe that $2^{n_2} = O(1.29153^n)$.

For a set $Y \subseteq V$, denote by $N(Y) \subseteq V$ the set of vertices that are adjacent to at least one vertex in Y.

Our strategy is to count each independent set Y via its parts $Y \cap V_1$ and $Y \cap V_2$ using the following elementary observation:

Lemma 2. *A set of vertices $Y \subseteq V$ is independent in G if and only if $Y \cap V_2 \subseteq V_2 \setminus N(Y \cap V_1)$ and both $Y \cap V_1$ and $Y \cap V_2$ are independent in G.*



In particular, we observe that $Y \cap V_2$ is oblivious to $Y \cap V_1$ except for the requirement $Y \cap V_2 \subseteq V_2 \setminus N(Y \cap V_1)$. Thus, in terms of space usage, we can collect together all eligible independent sets $Y_1 \subseteq V_1$ that share the set $Z_2 = V_2 \setminus N(Y_1)$. The function h in the following algorithm collects such sets Y_1 by their size $|Y_1|$ into the coefficients of the polynomial $h(Z_2)$. The function h requires only $O^*(2^{n_2})$ space, which in effect enables us to compute $i(X)$ for all $X \subseteq V$ in time $O^*(2^n)$ and less space than the number of independent sets in G. That is, using Lemma 2 we can beat the bound in Theorem 1.

In precise terms, we execute the following algorithm to obtain the number of proper k-colorings of G:

1. Set $r \leftarrow 0$.
2. For each $X_1 \subseteq V_1$ do:
   (a) For each $Z_2 \subseteq V_2$, set $h(Z_2) \leftarrow 0$.
   {{ The array h takes space $O^*(2^{n_2})$. }}
   (b) For each $Y_1 \subseteq X_1$, set

$$h(V_2 \setminus N(Y_1)) \leftarrow h(V_2 \setminus N(Y_1)) + z^{|Y_1|}[Y_1 \text{ is independent in } G].$$

   {{ This loop takes time $O^*(2^{|X_1|})$. }}
   (c) For each $Y_2 \subseteq V_2$, set $\ell(Y_2) \leftarrow z^{|Y_2|}[Y_2$ is independent in $G]$.
   {{ The array ℓ takes space $O^*(2^{n_2})$. }}
   (d) Set $g \leftarrow (h\zeta') \cdot \ell$.
   {{ Here "$\cdot$" denotes the elementwise product of two arrays.
   Time and space $O^*(2^{n_2})$ using the fast up-zeta transform on 2^{V_2}. }}
   (e) Set $j \leftarrow g\zeta$.
   {{ Time and space $O^*(2^{n_2})$ using the fast zeta transform on 2^{V_2}. }}
   (f) For each $X_2 \subseteq V_2$, set $r \leftarrow r + (-1)^{n-|X_1|-|X_2|} j(X_2)^k$.
3. Return the coefficient of z^n in r.

We establish the correctness of the algorithm by analyzing the contents of the array i during each iteration of the loop over $X_1 \subseteq V_1$:

Lemma 3. *For each fixed $X_1 \subseteq V_1$, we have $j(X_2) = i(X_1 \cup X_2)$ for all $X_2 \subseteq V_2$.*

Proof. Expanding the assignments in the algorithm and using Lemma 2, we have

$$\begin{aligned}
j(X_2) &= \sum_{Y_2 \subseteq X_2} g(Y_2) \\
&= \sum_{Y_2 \subseteq X_2} h\zeta'(Y_2)\ell(Y_2) \\
&= \sum_{Y_2 \subseteq X_2} \sum_{Y_2 \subseteq Z_2} h(Z_2)\ell(Y_2) \\
&= \sum_{Y_2 \subseteq X_2} \sum_{Y_2 \subseteq Z_2} \sum_{Y_1 \subseteq X_1} z^{|Y_1|}[Y_1 \text{ is independent in } G \text{ and } Z_2 = V_2 \setminus N(Y_1)] \\
&\qquad\qquad \times z^{|Y_2|}[Y_2 \text{ is independent in } G] \\
&= \sum_{Y_1 \subseteq X_1} \sum_{Y_2 \subseteq X_2} z^{|Y_1 \cup Y_2|}[Y_1 \cup Y_2 \text{ is independent in } G] \\
&= i(X_1 \cup X_2)\,.
\end{aligned}$$

□

To analyze the running time, we observe that the total running time (over all iterations $X_1 \subseteq V_1$) spent in Step 2(b) of the algorithm is within a polynomial factor of

$$\sum_{X_1 \subseteq V_1} 2^{|X_1|} = \sum_{j=0}^{n_1} \binom{n_1}{j} 2^j = 3^{n_1} = O(2^n)\,.$$

Thus, the total running time is $O^*(2^n)$, using space $O^*(2^{n_2}) = O(1.2916^n)$.

Because the computations are independent for each $X_1 \subseteq V_1$, they can be executed in parallel on $O^*(2^{n_1}) = O(1.5486^n)$ processors. While the space requirement per processor is $O(1.2916^n)$ and the total running time remains $O^*(2^n)$, the time requirement per processor varies, ranging from $O(1.5486^n)$ to $O(1.2916^n)$. This bottleneck can be removed by taking a more balanced scheme with n_1 and n_2 about equal, yielding time and space $O^*(2^{n/2})$ for each of the $O^*(2^{n/2})$ processors.

References

1. Bellman, R.: Combinatorial Processes and Dynamic Programming. In: Bellman, R., Hall, M. (eds.) Combinatorial Analysis, Proceedings of Symposia in Applied Mathematics, vol. 10, pp. 217–249. American Mathematical Society, Providence (1960)
2. Bellman, R.: Dynamic Programming Treatment of the Travelling Salesman Problem. J. Assoc. Comput. Mach. 9, 61–63 (1962)
3. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion–exclusion. SIAM J. Comput. Special Issue for FOCS 2006 39, 546–563 (2009)
4. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets Möbius: fast subset convolution. In: 39th ACM Symposium on Theory of Computing (STOC 2007), pp. 67–74. ACM Press, New York (2007)

5. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Computing the Tutte polynomial in vertex-exponential time. In: 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008), pp. 677–686. IEEE Computer Society Press, Los Alamitos (2008)
6. Fomin, F.V., Grandoni, F., Pyatkin, A., Stepanov, A.: Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications. ACM Transactions on Algorithms 5, 1–17 (2008)
7. Held, M., Karp, R.M.: A Dynamic Programming Approach to Sequencing Problems. J. Soc. Indust. Appl. Math. 10, 196–210 (1962)
8. Kennes, R.: Computational aspects of the Moebius transform of a graph. IEEE Transactions on Systems, Man, and Cybernetics 22, 201–223 (1991)
9. Koivisto, M., Parviainen, P.: A space-time tradeoff for permutation problems. In: 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), pp. 484–492 (2010)
10. Moon, J.W., Moser, L.: On cliques in graphs. Israel Journal of Mathematics 3, 23–28 (1965)

Testing 2-Vertex Connectivity and Computing Pairs of Vertex-Disjoint *s*-*t* Paths in Digraphs*

Loukas Georgiadis

Department of Informatics and Telecommunications Engineering,
University of Western Macedonia, Greece
lgeorg@uowm.gr

Abstract. We present an $O(m+n)$-time algorithm that tests if a given directed graph is 2-vertex connected, where m is the number of arcs and n is the number of vertices. Based on this result we design an $O(n)$-space data structure that can compute in $O(\log^2 n)$ time two internally vertex-disjoint paths from s to t, for any pair of query vertices s and t of a 2-vertex connected directed graph. The two paths can be reported in additional $O(k)$ time, where k is their total length.

1 Introduction

A directed (undirected) graph is *k-vertex connected* if it has at least $k+1$ vertices and the removal of any set of at most $k-1$ vertices leaves the graph strongly connected (connected). The *vertex connectivity* $\kappa \equiv \kappa(G)$ of a graph G is the maximum k such that G is k-vertex connected. Graph connectivity is one of the most fundamental concepts in graph theory with numerous practical applications [3]. Currently, the fastest known algorithm for computing κ is due to Gabow [12], with $O((n+\min\{\kappa^{5/2}, \kappa n^{3/4}\})m)$ running time. In [12], Gabow also showed how to test $\alpha\delta$-vertex connectivity in $O((\kappa+\sqrt{n})\sqrt{n}m)$ time, where δ is the minimum degree of the given graph and α is an arbitrary fixed constant less than one. Henzinger et al. [18] showed how to test k-vertex connectivity in time $O(\min\{k^3+n, kn\}m)$. They also gave a randomized algorithm for computing κ with error probability $1/2$ in time $O(nm)$. For an undirected graph, a result of Nagamochi and Ibaraki [21] allows m to be replaced by κn or kn in the above bounds. Cheriyan and Reif [9] showed how to test k-vertex connectivity in a directed graph with a Monte Carlo algorithm with running time $O((M(n)+nM(k))\log n)$ and error probability $< 1/n$, and with a Las Vegas algorithm with expected running time $O((M(n)+nM(k))k)$. In these bounds, $M(n)$ is the time to multiply two $n \times n$ matrices, which is $O(n^{2.376})$ [10].

Note that for constant κ or k the above bounds are $O(nm)$ for deterministic algorithms and $O(M(n))$ for randomized algorithms. To the best of our knowledge, these are also the best previously known bounds for testing $k = 2$ for a

* This research project has been funded by the John S. Latsis Public Benefit Foundation. The sole responsibility for the content of this paper lies with its author.

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 738–749, 2010.

directed graph. In Section 2 we present a linear-time algorithm for this problem. In the undirected case, linear-time algorithms were given by Tarjan [23] for testing $k = 2$, and by Hopcroft and Tarjan [19] for testing $k = 3$. Our algorithm is based on a new characterization of 2-vertex connected directed graphs, which utilizes the concept of *dominators in flowgraphs*. A flowgraph $G(s) = (V, A, s)$ is a directed graph with a distinguished source vertex $s \in V$ such that every vertex is reachable from s. The *dominance relation* in $G(s)$ is defined as follows: A vertex w *dominates* a vertex v if every path from s to v includes w. We denote by $dom(v)$ the set of vertices that dominate v. Obviously, $dom(s) = \{s\}$ and $dom(v) \supseteq \{s, v\}$, for any $v \neq s$; s and v are the *trivial dominators* of v. The dominance relation is transitive and its transitive reduction is the *dominator tree* D, which is rooted at s and satisfies the following property: For any two vertices v and w, w dominates v if and only if w is an ancestor of v in D [1]. For any vertex $v \neq s$, the *immediate dominator* of v is the parent of v in D. It is the unique vertex that dominates v and is dominated by all vertices in $dom(v) \setminus \{v\}$. The computation of dominators appears in several application areas, such as program optimization and code generation, constraint programming, circuit testing, and theoretical biology [17]. Lengauer and Tarjan [20] presented an $O(m\alpha(m, n))$-time algorithm for computing dominators, where $\alpha(m, n)$ is a very slow-growing functional inverse of Ackermann's function. This algorithm also works very well in practice [17], even though it has some conceptual complexities. There are even more complicated truly linear-time algorithms that run on random-access machines [2,6] and on pointer machines [15,14,5]. Our 2-vertex connectivity algorithm needs to test whether certain flowgraphs, derived from the input directed graph, have trivial dominators only (i.e., the immediate dominator of all vertices is the source vertex of the flowgraph). This can be done by computing the dominators of the flowgraph, but for our purpose a simpler alternative is to use a dominator-verification algorithm [16]. The algorithm given in [16] requires a linear-time solution to a special case of the disjoint set union problem [13] in order to achieve linear running time. With a standard disjoint set union structure the algorithm has running time $O(m\alpha(m, n))$ [24], and avoids the complexities of the Lengauer-Tarjan algorithm for computing dominators.

The second part of the paper (Section 3) deals with the task of computing two internally vertex-disjoint s-t paths (i.e., paths directed from s to t) in a 2-vertex connected directed graph, for any given source vertex s and target vertex t. This problem can be reduced to computing two edge-disjoint paths (by applying a standard vertex splitting procedure), which in turn can be carried out in $O(m)$ time by computing two flow-augmenting paths [3]. In Section 3 we present a faster algorithm for 2-vertex connected directed graphs. First we note that our linear-time algorithm for testing 2-vertex connectivity allows us to find, in linear time, a 2-vertex connected spanning subgraph of the input directed graph with $O(n)$ arcs. Hence, the flow-augmenting algorithm can compute two internally vertex-disjoint s-t paths in $O(n)$ time. We can improve this further with the use of *independent branchings*. A branching of a directed graph G is a rooted spanning tree of G such that all vertices other than the root have in-degree one,

whereas the root has in-degree zero. Two branchings of G are independent if for each vertex v, the two root-to-v paths are internally vertex-disjoint. In [16], a linear-time algorithm that constructs two independent branchings rooted at the same vertex was presented. Based on this result, we construct an $O(n)$-space data structure that can compute two internally vertex-disjoint s-t paths, for any s and t, in $O(\log^2 n)$ time, so that the two paths can be reported in constant time per vertex.

2 Testing 2-Vertex Connectivity

Let $G = (V, A)$ be the input directed graph (digraph). For any vertex $s \in V$, we denote by $G(s) = (V, A, s)$ the corresponding flowgraph with source vertex s. We can assume that G is strongly connected, which implies that all vertices are reachable from s and reach s. We also let $G^r(s)$ be the flowgraph derived from $G(s)$ after reversing all arc directions. For any $u, v \in V$, the *local connectivity* $\kappa(u, v)$ of G is defined as the maximum number of internally vertex-disjoint paths from u to v. By Menger's theorem (see, e.g., [3]) this is equal to the minimum number of cut vertices in an u-v separator if $(u, v) \notin A$. The next lemma relates local and global connectivity.

Lemma 1. *(See, e.g., [3])* $\kappa(G) = \min_{u,v \in V} \kappa(u, v)$.

Thus, a 2-vertex connected digraph $G = (V, A)$ satisfies the following property.

Lemma 2. *Let $G = (V, A)$ be a 2-vertex connected digraph. Then, for any vertex $s \in V$, both flowgraphs $G(s)$ and $G^r(s)$ have trivial dominators only.*

Proof. Lemma 1 and the fact that $\kappa(G) \geq 2$ imply that, for any vertex $v \in V - s$, there are at least two internally vertex-disjoint paths from s to v. Hence s is the immediate dominator of v in $G(s)$. Similarly, there are at least two internally vertex-disjoint paths from v to s. Hence s is also the immediate dominator of v in $G^r(s)$. □

Our goal is to prove that the following property characterizes 2-vertex connected digraphs.

Property 1. Let a and b be two distinct vertices of a digraph G. Then, all four flowgraphs $G(a)$, $G(b)$, $G^r(a)$ and $G^r(b)$ have trivial dominators only.

2.1 Dominators and 2-Vertex Connectivity

Lemma 2 implies that Property 1 is necessary for a digraph to be 2-vertex connected. Therefore, it remains to show that Property 1 is also sufficient. First we introduce some additional notation. We denote by $P[a, b]$ a simple path (i.e., a path with no repeated vertices) from a to b, and let $P(a, b]$ denote this path excluding a. Similarly, $P[a, b)$ excludes b, and $P(a, b)$ excludes both a and b. We define the *rank* of a vertex $c \in P[a, b]$ as the number of vertices in $P[a, c]$.

(There is no ambiguity in this definition since $P[a,b]$ is a simple path.) Let $P[x,y]$ and $Q[y,z]$ be two simple paths. We denote by $P[x,y] \cdot Q[y,z]$ the catenation of these two paths (which is not necessarily a simple path). Furthermore, let $R = P[x,y] \cdot Q[y,z]$. We denote by $R[x,z]$ a simple path from x to z that can be formed from R. One way to accomplish this is as follows. We find the vertex $w \in P[x,y] \cap Q[y,z]$ with the highest rank in $Q[y,z]$. Then we let $R[x,z] = P[x,w] \cdot Q[w,z]$.

For our construction we will need a series of technical lemmas.

Lemma 3. *([16]) Consider the flowgraph $G(s) = (V, A, s)$ and a vertex $v \neq s$, such that s is the immediate dominator of v in $G(s)$. If $(s,v) \notin A$ then there are two internally vertex-disjoint paths from s to v.*

Lemma 4. *Let $G = (V, A)$ be a digraph such that for each $(u,v) \notin A$ there are two internally vertex-disjoint paths from u to v. Then G is 2-vertex connected.*

Proof. By Lemma 1 we need to show that there is a pair of vertex-disjoint paths between any pair $u, v \in V$, so we consider the case $(u,v) \in A$. Observe that for any $w \in V \setminus \{u,v\}$ there is a path $P[u,w]$ that does not contain v. This is clear when $(u,w) \in A$. If $(u,w) \notin A$ then there are two vertex-disjoint paths from u to w, so they cannot both contain v. Similarly, we have a path $Q[w,v]$ that does not contain u. Therefore (u,v) and $P[u,w] \cdot Q[w,v]$ is a pair of internally vertex-disjoint paths from u to v. □

Lemma 5. *Consider three distinct vertices a,c and d such that the following two pairs of internally vertex-disjoint paths exist: $P_1[a,c]$ and $P_2[a,c]$, and $P_3[d,a]$ and $P_4[d,a]$. If $P_3[d,a) \cap P_1(a,c] \neq \emptyset$ then there are two internally vertex-disjoint paths from d to c.*

Proof. Let x be the vertex with the lowest rank in $P_3[d,a]$ such that $x \in P_3[d,a) \cap \big(P_1(a,c] \cup P_2(a,c]\big)$. (The premises of the lemma imply that x exists.) Furthermore, we can assume that $x \in P_3[d,a) \cap P_1(a,c]$, since we can alternate the role of $P_1[a,c]$ and $P_2[a,c]$ if $x \notin P_1(a,c]$. This also implies that $d \notin P_2[a,c]$. We distinguish the following cases:

a) $P_4(d,a) \cap P_1(a,c) = \emptyset$ and $P_4(d,a) \cap P_2(a,c) = \emptyset$. Let $R = P_4[d,a] \cdot P_2[a,c]$. Then the paths $P_3[d,x] \cdot P_1[x,c]$ and $R[d,c]$ are internally vertex-disjoint.[1]

b) $P_3(d,a) \cap P_2(a,c) = \emptyset$ and $P_4(d,a) \cap P_2(a,c) = \emptyset$. Suppose $P_4(d,a) \cap P_1(a,c) \neq \emptyset$, otherwise we have case (a). Let e be the vertex with the highest rank in $P_1[a,c]$ such that $e \in P_3[d,a) \cap P_1(a,c]$. Also let f be the vertex with the highest rank in $P_1[a,c]$ such that $f \in P_4(d,a) \cap P_1(a,c]$. Since $P_3[d,a)$ and $P_4(d,a)$ are vertex-disjoint we have $e \neq f$. If e has higher rank in $P_1[a,c]$ than f then the paths $P_3[d,e] \cdot P_1[e,c]$ and $P_4[d,a] \cdot P_2[a,c]$ are internally vertex-disjoint. Otherwise, the paths $P_4[d,f] \cdot P_1[f,c]$ and $P_3[d,a] \cdot P_2[a,c]$ are internally vertex-disjoint.

c) $P_3(d,a) \cap P_2(a,c) \neq \emptyset$ and $P_4(d,a) \cap P_2(a,c) = \emptyset$. If $P_4(d,a) \cap P_1(a,c) = \emptyset$ then we have case (a), so suppose $P_4(d,a) \cap P_1(a,c) \neq \emptyset$. Let g be the vertex with the lowest rank in $P_3[d,a]$ such that $g \in P_3(d,a) \cap P_2(a,c]$. Also, let e be the

[1] Note that we can have $c \in P_4(d,a)$, in which case R is not a simple path.

vertex with the highest rank in $P_1[a,c]$ such that $e \in P_3[d,g] \cap P_1(a,c]$, and let f be the vertex with the highest rank in $P_1[a,c]$ such that $f \in P_4(d,a) \cap P_1(a,c]$. Suppose the rank of e in $P_1[a,c]$ is lower than that of f. Then the paths $P_3[d,g] \cdot P_2[g,c]$ and $P_4[d,f] \cdot P_1[f,c]$ are internally vertex-disjoint. If, on the other hand, the rank of e in $P_1[a,c]$ is higher than that of f, then the paths $P_3[d,e] \cdot P_1[e,c]$ and $P_4[d,a] \cdot P_2[a,c]$ are internally vertex-disjoint.

d) $P_3(d,a) \cap P_2(a,c) \neq \emptyset$ and $P_4(d,a) \cap P_2(a,c) \neq \emptyset$. Let y be the vertex with the lowest rank in $P_4[d,a]$ such that $y \in P_4[d,a) \cap \big(P_1(a,c] \cup P_2(a,c]\big)$. First suppose that $y \in P_2(a,c]$. Then the paths $P_3[d,x] \cdot P_1[x,c]$ and $P_4[d,y] \cdot P_2[y,c]$ are internally vertex-disjoint.

Now consider that $y \in P_1(a,c)$. Let g be the vertex with the lowest rank in $P_3[d,a]$ such that $g \in P_3(d,a) \cap P_2(a,c]$, and let h be the vertex with the lowest rank in $P_4[d,a]$ such that $h \in P_4(d,a) \cap P_2(a,c]$. Also, let e be the vertex with the highest rank in $P_1[a,c]$ such that $e \in P_3[d,g] \cap P_1(a,c]$, and let f be the vertex with the highest rank in $P_1[a,c]$ such that $f \in P_4(d,h] \cap P_1(a,c]$. Since $P_3[d,a)$ and $P_4(d,a)$ are vertex-disjoint we have $e \neq f$. If the rank of e in $P_1[a,c]$ is lower than that of f then the paths $P_3[d,g] \cdot P_2[g,c]$ and $P_4[d,f] \cdot P_1[f,c]$ are internally vertex-disjoint. Otherwise, the paths $P_3[d,e] \cdot P_1[e,c]$ and $P_4[d,h] \cdot P_2[h,c]$ are internally vertex-disjoint. □

Lemma 6. *Consider three distinct vertices a,b and c such that the following two pairs of internally vertex-disjoint paths exist: $P_1[a,c]$ and $P_2[a,c]$, and $P_3[b,c]$ and $P_4[b,c]$. Then there are two internally vertex-disjoint paths $Q[a,c]$ and $Q'[b,c]$.*

Proof. If $P_1[a,c) \cap P_3[b,c) = \emptyset$ then, clearly, the lemma holds with $Q[a,c] = P_1[a,c]$ and $Q'[b,c] = P_3[b,c]$. Now suppose that $P_3[b,c)$ intersects $P_1[a,c)$. Let e be the vertex with the lowest rank in $P_3[b,c]$ such that $e \in P_3[b,c) \cap \big(P_1[a,c) \cup P_2[a,c)\big)$. We can assume, with no loss of generality, that $e \in P_1[a,c)$. If $e = a$ then the paths $Q[a,c] = P_3[a,c]$ and $Q'[b,c] = P_4[b,c]$ are internally vertex-disjoint. Otherwise, if $e \neq a$, the paths $Q[a,c] = P_2[a,c]$ and $Q'[b,c] = P_3[b,e] \cdot P_1[e,c]$ are internally vertex-disjoint. □

Lemma 7. *Let a, b be any two distinct vertices of a digraph $G = (V,A)$ that satisfy Property 1. Then for any vertex $c \notin \{a,b\}$ there are two internally vertex-disjoint paths $Q[a,c]$ and $Q'[b,c]$.*

Proof. Property 1 and Lemma 3 imply that for any $c \neq a$, if $(a,c) \notin A$ then there are two internally vertex-disjoint paths from a and c. Similarly, for any $c \neq b$, if $(b,c) \notin A$ then there are two internally vertex-disjoint paths from b and c. It is clear that the lemma holds when G contains both arcs (a,c) and (b,c). Next consider that G contains the arc (a,c) but not (b,c). Then there are two internally vertex-disjoint paths from b and c, therefore they cannot both contain a and the lemma follows. The case $(a,c) \notin A$ and $(b,c) \in A$ is symmetric. Finally, assume that $(a,c) \notin A$ and $(b,c) \notin A$. Now we have two internally vertex-disjoint

paths from a to c and two internally vertex-disjoint paths from b to c, hence the result follows from Lemma 6. □

Symmetrically, we have the next two statements regarding paths entering a and b.

Lemma 8. *Consider three distinct vertices a,b and d such that the following two pairs of internally vertex-disjoint paths exist: $P_1[d,a]$ and $P_2[d,a]$, and $P_3[d,b]$ and $P_4[d,b]$. Then there are two internally vertex-disjoint paths $Q[d,a]$ and $Q'[d,b]$.*

Lemma 9. *Let a, b be any two distinct vertices of a digraph $G=(V,A)$ that satisfy Property 7. Then for any vertex $d \notin \{a,b\}$ there are two internally vertex-disjoint paths $Q[d,a]$ and $Q'[d,b]$.*

Before proceeding to our main lemma we make the following observation. Our goal is to show that for any pair of vertices d and c there are two internally vertex-disjoint paths from d to c. Unfortunately we cannot obtain the desired result immediately by applying Lemmas 7 and 9; Lemma 7 gives us two paths, from a and b to c, that only meet at c. Lemma 9 gives us two paths, from d to a and b, that only meet at d. These paths, however, do not suffice to construct two internally vertex-disjoint paths from d to c. Therefore our arguments need to be more subtle.

Lemma 10. *Let a, b be any two distinct vertices of a digraph $G=(V,A)$ that satisfy Property 7. Then G is 2-vertex connected.*

Proof. In light of Lemma 4 it suffices to show that for any pair $c,d \in V$, G contains the arc (d,c) or two internally vertex-disjoint paths from d to c. This follows immediately from Lemma 3 when $d \in \{a,b\}$ or $c \in \{a,b\}$. Now consider $d \notin \{a,b\}$ and $c \notin \{a,b\}$. We will exhibit two internally vertex-disjoint paths from d to c. To that end, we distinguish the following cases:

a) Suppose $(a,c) \in A$. By Lemma 9 we have two internally vertex-disjoint paths $Q[d,a]$ and $Q'[d,b]$, i.e., $Q(d,a] \cap Q'(d,b] = \emptyset$. In particular, note that $a \notin Q'[d,b]$ and $b \notin Q[d,a]$. First consider that also $(b,c) \in A$. If none of the paths $Q[d,a]$ and $Q'[d,b]$ contains c then $Q[d,a] \cdot (a,c)$ and $Q'[d,b] \cdot (b,c)$ are internally vertex-disjoint. If $Q[d,a]$ contains c then $Q'[d,b]$ does not, so $Q[d,c]$ and $Q'[d,b] \cdot (b,c)$ are internally vertex-disjoint. The case $c \in Q'[d,b]$ is symmetric.

If $(b,c) \notin A$ then there are two internally vertex-disjoint paths $P_1[b,c]$ and $P_2[b,c]$. Suppose $Q[d,a] \cap P_1[b,c] = \emptyset$. Let $R = Q[d,a] \cdot (a,c)$ and $R' = Q'[d,b] \cdot P_1[b,c]$. Then $R[d,c] (= R)$ and $R'[d,c]$ are internally vertex-disjoint. Now consider $Q[d,a] \cap P_1[b,c] \neq \emptyset$. We would like to apply Lemma 5 for vertices d, b and c but we need two internally vertex-disjoint paths from d to b. To that end, let us assume that $(a,b) \in A$ and set $Q''[d,b] = Q[d,a] \cdot (a,b)$. Then Lemma 5 provides two internally vertex-disjoint paths $R[d,c]$ and $R'[d,c]$. If none of these paths uses the arc (a,b) then we have found two internally vertex-disjoint paths from d to c in G. Otherwise, suppose that $R[d,c]$ contains (a,b). Then the paths $R[d,a] \cdot (a,c)$ and $R'[d,c]$ are internally vertex-disjoint.

The case $(b,c) \in A$ is symmetric.

b) The case $(d,a) \in A$ can be analyzed similarly to case (a) but we provide the details for completeness. From Lemma 7 we have two internally vertex-disjoint paths $Q[a,c]$ and $Q'[b,c]$, i.e., $Q[a,c) \cap Q'[b,c) = \emptyset$. In particular, note that $a \notin Q'[b,c]$ and $b \notin Q[a,c]$. First consider that also $(d,b) \in A$. If none of the paths $Q[a,c]$ and $Q'[b,c]$ contains d then $(d,a) \cdot Q[a,c]$ and $(d,b) \cdot Q'[b,c]$ are internally vertex-disjoint. If $Q[a,c]$ contains d then $Q'[b,c]$ does not, so $Q[d,c]$ and $(d,b) \cdot Q'[b,c]$ are internally vertex-disjoint. The case $d \in Q'[b,c]$ is symmetric.

If $(d,b) \notin A$ then there are two internally vertex-disjoint paths $P_1[d,b]$ and $P_2[d,b]$. Suppose $Q[a,c] \cap P_1[d,b] = \emptyset$. Let $R = (d,a) \cdot Q[a,c]$ and $R' = P_1[d,b] \cdot Q'[b,c]$. Then $R[d,c] (= R)$ and $R'[d,c]$ are internally vertex-disjoint. Now consider $Q[a,c] \cap P_1[d,b] \neq \emptyset$. We apply Lemma 5 for vertices d, b and c as in case (a). We assume at first that $(b,a) \in A$ and set $Q''[b,c] = (b,a) \cdot Q[a,c]$. Then Lemma 5 gives us two internally vertex-disjoint paths $R[d,c]$ and $R'[d,c]$. If none of these paths uses the arc (b,a) then we have found two internally vertex-disjoint paths from d to c in G. Otherwise, suppose that $R[d,c]$ contains (b,a). Then the paths $(d,a) \cdot R[a,c]$ and $R'[d,c]$ are internally vertex-disjoint.

The case $(d,b) \in A$ is symmetric.

c) It remains to examine the case where none of the arcs (a,c), (d,a), (b,c) and (d,b) is present. By Lemma 3 we have the following pairs of internally vertex-disjoint paths: $P_1[d,a]$ and $P_2[d,a]$, $P_3[a,c]$ and $P_4[a,c]$, $P_5[d,b]$ and $P_6[d,b]$, and $P_7[b,c]$ and $P_8[b,c]$. If $P_1[d,a)$ or $P_2[d,a)$ intersects $P_3(a,c]$ or $P_4(a,c]$ then Lemma 5 gives us two internally vertex-disjoint paths from d to c. Similarly, if $P_5[d,b)$ or $P_6[d,b)$ intersects $P_7(b,c]$ or $P_8(b,c]$ then Lemma 5 gives us two internally vertex-disjoint paths from d to c. Now we suppose that $P_1[d,a)$ and $P_2[d,a)$ do not intersect $P_3(a,c]$ and $P_4(a,c]$, and also that $P_5[d,b)$ and $P_6[d,b)$ do not intersect $P_7(b,c]$ and $P_8(b,c]$.

First we consider that $P_5[d,b]$ intersects $P_3[a,c] \cup P_4[a,c]$. Let f be the vertex with the lowest rank in $P_5[d,b]$ such that $f \in P_5[d,b] \cap \big(P_3[a,c] \cup P_4[a,c]\big)$. Without loss of generality we can assume that $f \in P_3[a,c]$. Suppose $f = a$. Then $a \notin P_6[d,b]$. If $P_6[d,b]$ intersects $P_3[a,c] \cup P_4[a,c]$ then we can alternate the role of $P_5[d,b]$ and $P_6[d,b]$ and consider the case $f \neq a$. So now let $P_6[d,b] \cap \big(P_3[a,c] \cup P_4[a,c]\big) = \emptyset$. Let h be the vertex with the lowest rank in $P_7[b,c]$ such that $h \in P_7[b,c] \cap \big(P_3[a,c] \cup P_4[a,c]\big)$. Without loss of generality, suppose $h \in P_3[a,c]$. Then the paths $P_5[d,a] \cdot P_4[a,c]$ and $P_6[d,b] \cdot P_7[b,h] \cdot P_3[h,c]$ are internally vertex-disjoint. Next, consider $f \neq a$. Let e be the vertex with the highest rank in $P_5[d,f]$ such that $e \in P_5[d,f] \cap \big(P_1[d,a] \cup P_2[d,a]\big)$. Note that $e \neq a$. (Also $e \neq f$). Without loss of generality, suppose $e \in P_1[d,a]$. Then $P_1[d,e] \cdot P_5[e,f] \cdot P_3[f,c]$ and $P_2[d,a] \cdot P_4[a,c]$ are internally vertex-disjoint.

The cases $P_6[d,b] \cap \big(P_3[a,c] \cup P_4[a,c]\big) \neq \emptyset$, $P_1[d,a] \cap \big(P_7[b,c] \cup P_8[b,c]\big) \neq \emptyset$, and $P_2[d,a] \cap \big(P_7[b,c] \cup P_8[b,c]\big) \neq \emptyset$ are treated similarly.

Finally suppose that $P_5[d,b]$ and $P_6[d,b]$ do not intersect $P_3[a,c]$ and $P_4[a,c]$, and also that $P_1[d,a]$ and $P_2[d,a]$ do not intersect $P_7[b,c]$ and $P_8[b,c]$. We apply Lemma 8 for a, b, and d and get two internally vertex-disjoint paths $Q_1[d,a]$ and $Q_2[d,b]$. Then we apply Lemma 6 for a, b, and c get two internally vertex-disjoint

paths $Q_3[a,c]$ and $Q_4[b,c]$. Since $\big(P_1[d,a] \cup P_2[d,a]\big) \cap \big(P_7[b,c] \cup P_8[b,c]\big) = \emptyset$ and $\big(P_5[d,b] \cup P_6[d,b]\big) \cap \big(P_3[a,c] \cup P_4[a,c]\big) = \emptyset$, we have $Q_1[d,a] \cap \big(Q_3(a,c] \cup Q_4[b,c]\big) = \emptyset$ and $Q_2[d,b] \cap \big(Q_3[a,c] \cup Q_4(b,c]\big) = \emptyset$.Thus $Q_1[d,a] \cdot Q_3[a,c]$ and $Q_2[d,b] \cdot Q_4[b,c]$ are internally vertex-disjoint. □

Combining Lemmas 2 and 10 we have:

Theorem 1. *Let a, b be two arbitrary but distinct vertices of a digraph G. Then G is 2-vertex connected if and only if it satisfies Property 1 for a and b.*

2.2 Linear-Time Algorithm

Based on Theorem 1 we propose the following algorithm for testing 2-vertex connectivity: Given the input digraph $G = (V, A)$, we first compute the reverse graph $G^r = (V, A^r)$, where $A^r = \{(x, y) \mid (y, x) \in A\}$. Then we pick two distinct vertices $a, b \in V$ and verify that for each $s \in \{a, b\}$ the flowgraphs $G(s)$ and $G^r(s)$ have trivial dominators only; we report that G is 2-vertex connected if and only if this is true. If the input graph is strongly connected but not 2-vertex connected, then we would like to report a cut vertex, i.e., a vertex whose removal increases the number of strongly connected components, as a certificate of < 2-vertex connectivity. To that end, we can use another property of the trivial-dominator-verification algorithm in [16]. Namely, if a flowgraph $G(s)$ has non-trivial dominators then the verification algorithm reports two vertices $x \neq s$ and y, such that x is the immediate dominator of y in $G(s)$. Thus, the removal of x destroys all paths from s to y, which implies that x is a cut vertex.[2]

The correctness of the above algorithm follows immediately from Theorem 1. We now turn to the running time. Computing G^r in $O(m + n)$ time is easy. Furthermore, we can test if a flowgraph has only trivial dominators in $O(m + n)$ time using the algorithm in [16]. (Alternatively, we can use a linear-time algorithm for computing the dominators of the flowgraph but this computation is more complicated [5].) Since our 2-vertex-connectivity algorithm uses four such tests, the total running time is $O(m+n)$. In practice, we can use a simpler $O(m\alpha(m, n))$-time version of the trivial-dominator-verification algorithm in [16], which uses a standard disjoint set union data structure [13] instead of the linear-time algorithm of Gabow and Tarjan [13].

Theorem 2. *There is a linear-time algorithm for testing 2-vertex connectivity of a digraph G. If G is strongly connected but not 2-vertex connected then the algorithm returns a cut vertex.*

3 Computing Two Vertex-Disjoint s-t Paths

We now consider the problem of preprocessing a 2-vertex connected digraph $G = (V, A)$ into a data structure that can efficiently compute two internally

[2] We remark that by using an algorithm for computing dominators (instead of just verifying) we can compute all the cut vertices of a strongly connected digraph.

vertex-disjoint paths from d to c, for any pair of distinct vertices $d, c \in V$. Our data structure is based on the proof of Theorem 1 and on a linear-time algorithm of [16], which computes two branchings T_1 and T_2 of a flowgraph $G(s) = (V, A, s)$. These are (directed) spanning trees of G, which are rooted at s and have the following property: For any vertex $v \in V$, the two directed paths from s to v in T_1 and T_2, denoted by $T_1[s, v]$ and $T_2[s, v]$ respectively, meet only at the dominators of v in $G(s)$. Therefore, if $G = (V, A)$ is 2-vertex connected then, by Lemma 2, $T_1[s, v]$ and $T_2[s, v]$ are internally vertex-disjoint; two branchings that have this property are called *independent*. We begin by computing the following pairs of independent branchings: T_1 and T_2 of $G^r(a)$, T_3 and T_4 of $G(a)$, T_5 and T_6 of $G^r(b)$, and T_7 and T_8 of $G(b)$, where a and b are two arbitrary but distinct vertices of G. Let A' be the set of arcs in these eight trees. Then, Theorem 1 implies that the digraph $G' = (V, A')$ is a 2-vertex connected spanning subgraph of G. Therefore, we can compute a pair of internally vertex-disjoint d-c paths in G'. This computation takes $O(n)$ time with a flow-augmenting algorithm (see, e.g., [3]), since A' has $O(n)$ arcs.

Next, we describe how to compute these paths in $O(\log^2 n)$ time, so that we can report them in additional $O(k)$ time, where k is the total length of the two paths. The proof of Theorem 1 finds two internally vertex-disjoint paths from d to c, using the following four pairs of internally vertex-disjoint paths: $P_1[d, a]$ and $P_2[d, a]$, $P_3[a, c]$ and $P_4[a, c]$, $P_5[d, b]$ and $P_6[d, b]$, and $P_7[b, c]$ and $P_8[b, c]$. In order to answer a query for two internally vertex-disjoint paths from d to c, we can use the corresponding paths on the branchings $T_1, \ldots, T_8$, i.e., we set $P_1[d, a] = (T_1[a, d])^r$, $P_2[d, a] = (T_2[a, d])^r$, $P_3[a, c] = T_3[a, c]$, $P_4[a, c] = T_4[a, c]$, $P_5[d, b] = (T_5[b, d])^r$, $P_6[d, a] = (T_6[a, d])^r$, $P_7[a, c] = T_7[a, c]$, $P_8[a, c] = T_8[a, c]$.

Let S_1 and S_2 be any two rooted trees on the same set of vertices. We define a set of operations on S_1 and S_2 that enable an efficient implementation of the construction given in Section 2.1. Consider four vertices x_1, y_1, x_2 and y_2, such that x_1 is an ancestor of y_1 in S_1 and x_2 is an ancestor of y_2 in S_2. We need a data structure that supports the following set of operations:

(i) Test if $S_1[x_1, y_1]$ contains x_2.
(ii) Return the topmost vertex in $S_1(x_1, y_1]$.
(iii) Test if $S_1[x_1, y_1]$ and $S_2[x_2, y_2]$ contain a common vertex.
(iv) Find the lowest ancestor of y_2 in $S_2[x_2, y_2]$ that is contained in $S_1[x_1, y_1]$.
(v) Find the highest ancestor of y_2 in $S_2[x_2, y_2]$ that is contained in $S_1[x_1, y_1]$.

By examining the construction of Section 2.1 we can verify that the above operations suffice for our needs. (We need a constant number of these operations per query.) For instance, we can find the vertex e with the highest rank in $P_5[d, f]$ such that $e \in P_5[d, f] \cap P_1[d, a]$ (refer to case (c) in the proof of Lemma 10) by applying operation (v) with $S_1 = T_1$, $S_2 = T_5$, $x_1 = a$, $y_1 = d$, $x_2 = f$ and $y_2 = d$. Operation (ii) is useful when we want to exclude the first vertex of a path in some computation. Excluding the last vertex on a path of S_j ($j \in \{1, 2\}$) is straightforward if we maintain a pointer from a node to its parent in S_j.

We develop an $O(n)$-space data structure that supports operations (i)-(v) efficiently. First note that operation (i) can be answered from (iii) if we set

$y_2 = x_2$. Furthermore, operation (iii) can be answered from (iv) or (v); if $S_1[x_1, y_1] \cap S_2[x_2, y_2] = \emptyset$ then these operations return null. Now it remains to implement operations (ii), (iv) and (v). We begin by assigning a depth-first search interval (as in [11]) to each vertex in S_1. Let $I_1(x) = [s_1(x), t_1(x)]$ be the interval of a vertex x in S_1; $s_1(x)$ is the time of the first visit to x (during the depth-first search) and $t_1(x)$ is the time of the last visit to x. (These times are computed by incrementing a counter after visiting or leaving a vertex during the search.) This way all the assigned $s_1()$ and $t_1()$ values are distinct, and for any vertex x we have $1 \leq s_1(x) < t_1(x) \leq 2n$. Moreover, by well-known properties of depth-first search, we have that x is an ancestor of y in S_1 if and only if $I_1(y) \subseteq I(x)$; if x and y are unrelated in S_1 then $I_1(x)$ and $I_1(y)$ are disjoint. Now, for operation (ii) we simply need to locate the child z of x_1 in S_1 such that $s_1(y_1) \in I_1(z)$. This is a static predecessor search problem that can be solved in $O(\log n)$ time with binary search (which suffices here) or in $O(\log \log n)$ time with more advanced structures [4].

In order to support operations (iii)-(v) efficiently we also assign a depth-first search interval $I_2(x) = [s_2(x), t_2(x)]$ to each vertex x in S_2. Next, we map each vertex x to an axis-parallel rectangle $R(x) = I_1(x) \times I_2(x)$. Let $\mathcal{R}$ be the set of all axis-parallel rectangles $R(x)$. We implement operations (iv) and (v) as ray shooting queries in the subdivision induced by $\mathcal{R}$. For any two vertices x and y, we define $R(x, y) = I_1(x) \times I_2(y)$. Then, $R(x) \equiv R(x, x)$. Consider two rectangles $R(x_1, x_2)$ and $R(y_1, y_2)$. If $I_1(x_1) \cap I_1(y_1) = \emptyset$ or $I_2(x_2) \cap I_2(y_2) = \emptyset$, then $R(x_1, x_2)$ and $R(y_1, y_2)$ do not intersect. Now suppose that both $I_1(x_1) \cap I_1(y_1) \neq \emptyset$ and $I_2(x_2) \cap I_2(y_2) \neq \emptyset$. Let $I_1(y_1) \subseteq I_1(x_1)$. If also $I_2(y_2) \subseteq I_2(x_2)$ then $R(y_1, y_2)$ is contained in $R(x_1, x_2)$; we denote this by $R(y_1, y_2) \subseteq R(x_1, x_2)$. Otherwise, if $I_2(x_2) \subseteq I_2(y_2)$ then both vertical edges of $R(y_1, y_2)$ intersect both horizontal edges of $R(x_1, x_2)$. Next, consider a rectangle $R(x_1, x_2)$ and let $\mathcal{R}(x_1, x_2) = \{R(z) \in \mathcal{R} \ : \ R(x_1, x_2) \subseteq R(z)\}$, i.e., the rectangles in $\mathcal{R}$ containing $R(x_1, x_2)$. The properties of the intervals $I_1()$ and $I_2()$ imply that we can order the rectangles in $\mathcal{R}(x_1, x_2)$ with respect to their vertical distance from $R(x_1, x_2)$. More formally, let $R(z_1), R(z_2), \ldots, R(z_\xi)$ be the rectangles in $\mathcal{R}(x_1, x_2)$ ordered by increasing $t_2(z_j)$ (the height of the upper horizontal edge). Also, let $R(z_{i_1}), R(z_{i_2}), \ldots, R(z_{i_\xi})$ be the rectangles in $\mathcal{R}(x_1, x_2)$ ordered by decreasing $s_2(z_{i_j})$ (the height of the lower horizontal edge). Then $i_j = j$, $j = 1, \ldots, \xi$. Now let $Q = R(y_1, y_2)$ and let $Q' = R(x_1, x_2)$. To perform operation (iv) we locate the rectangle $R(z) \in \mathcal{R}(y_1, y_2)$ with minimum $t_2(z)$ (and maximum $s_2(z)$). For operation (v) we locate the rectangle $R(z) \in \mathcal{R}(y_1, y_2)$ with maximum $t_2(z)$ (and minimum $s_2(z)$) such that $R(z) \subseteq Q'$. We can perform these operations efficiently by adapting a data structure of Chazelle [8]. The data structure consists of a binary search tree T on the vertical coordinates $s_2()$ and $t_2()$ of the vertices, and is constructed as follows. Let ℓ be an infinite horizontal line with $|\mathcal{R}|$ horizontal rectangle edges on each side. This line partitions $\mathcal{R}$ into three subsets: $\mathcal{R}_\uparrow$ contains the rectangles completely above ℓ, $\mathcal{R}_\downarrow$ contains the rectangles completely bellow ℓ, and $\mathcal{R}_\ell$ contains the rectangles intersecting ℓ. We associate with the root r of T the set $\mathcal{R}_\ell$. The left (resp. right) subtree of r is

defined recursively for the set $\mathcal{R}_\downarrow$ (resp. $\mathcal{R}_\uparrow$). Clearly, T has $O(\log n)$ height. For any node $v \in T$, we let $\mathcal{R}(v)$ denote the set of rectangles associated with v.

Let $q = (s_1(y_1), s_2(y_2))$, i.e., the lower left corner of Q. (Any corner of Q will do as well.) We implement operation (iv) as a ray shooting query in the subdivision induced by $\mathcal{R}(v_i)$, for each node $v_i \in T$ on the path $(v_0, v_1, \ldots, v_h)$ from $r = v_0$ to the leaf v_h that corresponds to the vertical coordinate $s_2(y_2)$. Suppose that the horizontal line $\ell(v_i)$ associated with node v_i is above q. Then, we locate the first rectangle $R_q^i \in \mathcal{R}(v_i)$ that is intersected by the vertical ray $[q, (s_1(y_1), -\infty)]$. If $\ell(v_i)$ is below q then we locate the first rectangle $R_q^i \in \mathcal{R}(v_i)$ that is intersected by the vertical ray $[q, (s_1(y_1), +\infty)]$. In either case, R_q^i can be found in $O(\log n)$ time using a planar point location data structure [22]. The answer to query (iv) is the rectangle $R(z) \in \{R_q^0, R_q^1, \ldots, R_q^h\}$ with minimum $t_2(z)$, therefore it can be found in total $O(h \log n) = O(\log^2 n)$ time. Operation (v) is carried out similarly. Let $q'_s = (s_1(y_1), s_2(x_2))$ and $q'_t = (s_1(y_1), t_2(x_2))$, respectively, be the projection of q on the lower and upper edge of Q'. Again we perform a ray shooting query in the subdivision induced by $\mathcal{R}(v_i)$, for each node $v_i \in T$ on the path $(v_0, v_1, \ldots, v_h)$ from $r = v_0$ to the leaf v_h that corresponds to the vertical coordinate $s_2(y_2)$. Suppose that the horizontal line $\ell(v_i)$ associated with node v_i is above q. Then, we locate the first rectangle $R_q^i \in \mathcal{R}(v_i)$ that is intersected by the vertical ray $[q'_s, (s_1(y_1), +\infty)]$. If $\ell(v_i)$ is below q then we locate the first rectangle $R_q^i \in \mathcal{R}(v_i)$ that is intersected by the vertical ray $[q'_t, (s_1(y_1), -\infty)]$. The answer to query (v) is the rectangle $R(z) \in \{R_q^0, R_q^1, \ldots, R_q^h\}$ with maximum $t_2(z)$ such that $Q \subseteq R(z) \subseteq Q'$. Therefore, it can be found in total $O(\log^2 n)$ time. It is easy to verify that the space bound for the above data structure is $O(n)$. For the construction of Section 2.1 we actually need such a data structure for several pairs of the branchings $T_1, \ldots, T_8$, but the total space is still $O(n)$.

Theorem 3. *Let $G = (V, A)$ be a 2-vertex connected digraph $G = (V, A)$ with n vertices. We can construct an $O(n)$-space data structure that can compute two internally vertex-disjoint paths from d to c in $O(\log^2 n)$ time, for any two distinct vertices $d, c \in V$. The two paths can be reported in additional $O(k)$ time, where k is their total length.*

Finally, we remark that the query time can be reduced to $O(\log n \sqrt{\log n / \log\log n})$ by applying the result of [7].

References

1. Aho, A.V., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools. Addison-Wesley, Reading (1986)
2. Alstrup, S., Harel, D., Lauridsen, P.W., Thorup, M.: Dominators in linear time. SIAM Journal on Computing 28(6), 2117–2132 (1999)
3. Bang-Jensen, J., Gutin, G.: Digraphs: Theory, Algorithms and Applications (Springer Monographs in Mathematics), 1st edn., 3rd printing edn. Springer, Heidelberg (2002)
4. Beame, P., Fich, F.E.: Optimal bounds for the predecessor problem and related problems. J. Comput. Syst. Sci. 65(1), 38–72 (2002)

5. Buchsbaum, A.L., Georgiadis, L., Kaplan, H., Rogers, A., Tarjan, R.E., Westbrook, J.R.: Linear-time algorithms for dominators and other path-evaluation problems. SIAM Journal on Computing 38(4), 1533–1573 (2008)
6. Buchsbaum, A.L., Kaplan, H., Rogers, A., Westbrook, J.R.: A new, simpler linear-time dominators algorithm. ACM Transactions on Programming Languages and Systems 20(6), 1265–1296 (1998); Corrigendum appeared 27(3), 383-387 (2005)
7. Chan, T.M., Patraşcu, M.: Transdichotomous results in computational geometry, i: Point location in sublogarithmic time. SIAM Journal on Computing 39(2), 703–729 (2009)
8. Chazelle, B.: Filtering search: A new approach to query-answering. SIAM Journal on Computing 15(3), 703–724 (1986)
9. Cheriyan, J., Reif, J.H.: Directed s-t numberings, rubber bands, and testing digraph k-vertex connectivity. Combinatorica, 435–451 (1994)
10. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. J. Symb. Comput. 9(3), 251–280 (1990)
11. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stei, C.: Introduction to Algorithms, 2nd edn. The MIT Press, Cambridge (2001)
12. Gabow, H.N.: Using expander graphs to find vertex connectivity. Journal of the ACM 53(5), 800–844 (2006)
13. Gabow, H.N., Tarjan, R.E.: A linear-time algorithm for a special case of disjoint set union. Journal of Computer and System Sciences 30(2), 209–221 (1985)
14. Georgiadis, L.: Linear-Time Algorithms for Dominators and Related Problems. PhD thesis, Princeton University (2005)
15. Georgiadis, L., Tarjan, R.E.: Finding dominators revisited. In: Proc. 15th ACM-SIAM Symp. on Discrete Algorithms, pp. 862–871 (2004)
16. Georgiadis, L., Tarjan, R.E.: Dominator tree verification and vertex-disjoint paths. In: Proc. 16th ACM-SIAM Symp. on Discrete Algorithms, pp. 433–442 (2005)
17. Georgiadis, L., Tarjan, R.E., Werneck, R.F.: Finding dominators in practice. Journal of Graph Algorithms and Applications (JGAA) 10(1), 69–94 (2006)
18. Henzinger, M.R., Rao, S., Gabow, H.N.: Computing vertex connectivity: New bounds from old techniques. Journal of Algorithms 34, 222–250 (2000)
19. Hopcroft, J.E., Tarjan, R.E.: Dividing a graph into triconnected components. SIAM Journal on Computing 2(3), 135–158 (1973)
20. Lengauer, T., Tarjan, R.E.: A fast algorithm for finding dominators in a flowgraph. ACM Transactions on Programming Languages and Systems 1(1), 121–141 (1979)
21. Nagamochi, H., Ibaraki, T.: A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. Algorithmica 7, 583–596
22. Sarnak, N., Tarjan, R.E.: Planar point location using persistent search trees. Communications of the ACM 29(7), 669–679 (1986)
23. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM Journal on Computing 1(2), 146–159 (1972)
24. Tarjan, R.E.: Edge-disjoint spanning trees and depth-first search. Acta Informatica 6(2), 171–185 (1976)

Author Index

Achilleos, Antonis II-369
Alistarh, Dan II-115
Ambos-Spies, Klaus I-503
Amir, Amihood I-43
Anagnostopoulos, Aris I-114
Applebaum, Benny I-152
Atserias, Albert I-102

Bakibayev, Timur I-503
Bansal, Nikhil I-250, I-287, I-324, II-273
Bateni, MohammadHossein I-67
Beimel, Amos I-451
Ben Daniel, Sebastian I-451
Berenbrink, Petra II-127
Bhattacharya, Amitava I-438
Björklund, Andreas I-727
Blasiak, Anna II-100
Blocki, Jeremiah II-393
Blum, Norbert II-163
Bojańczyk, Mikołaj I-515
Boker, Udi II-76
Bollig, Benedikt II-587
Bonichon, Nicolas I-19
Borodin, Allan I-90
Braud, Laurent II-88
Brázdil, Tomáš II-478, II-539
Briët, Jop I-31
Buchbinder, Niv I-287
Bunn, Paul II-236

Cai, Jin-Yi I-275
Carayol, Arnaud II-88
Chalopin, Jérémie II-515
Chambart, Pierre II-64
Chandran, Nishanth II-249
Chan, T-H. Hubert II-405
Chatterjee, Krishnendu II-599
Chechik, Shiri II-261
Chen, Ning II-418
Chen, Xi I-275
Chen, Yijia II-321
Cheraghchi, Mahdi I-552
Christodoulou, George II-430
Cicalese, Ferdinando I-527
Clementi, Andrea E.F. II-490
Coecke, Bob II-297
Colcombet, Thomas II-563
Cole, Richard I-226
Collins, Andrew II-502
Czyzowicz, Jurek II-127, II-502

Dalgaard Larsen, Kasper I-605
DasGupta, Bhaskar I-438
Das, Shantanu II-515
Dell, Holger I-426
Deng, Xiaotie II-418
de Oliveira Filho, Fernando Mário I-31
Dietzfelbinger, Martin I-213
Doyen, Laurent II-599
Duan, Ran I-201
Dumrauf, Dominic I-1
Duncan, Ross II-285

Eisenberg, Estrella I-43
Eisenbrand, Friedrich I-299
Elsässer, Robert II-127
Emek, Yuval II-261
Epstein, Leah I-336
Esparza, Javier II-539
Even, Guy II-139

Fakcharoenphol, Jittat I-176
Fearnley, John II-551
Flum, Jörg II-321
Fontaine, Gaëlle II-381
Fountoulakis, Nikolaos I-348
Fraigniaud, Pierre II-1

Gamzu, Iftah I-582
Garay, Juan II-249
Gastin, Paul II-587
Gavoille, Cyril I-19
Gehrke, Mai II-151
Genest, Blaise II-52
Georgiadis, Loukas I-738
Giacobazzi, Roberto II-211
Gilbert, Seth II-115
Gimbert, Hugo II-52, II-527
Gąsieniec, Leszek II-127, II-502
Goerdt, Andreas I-213

Goldberg, Leslie Ann I-396
Göller, Stefan II-575
Goubault-Larrecq, Jean II-2
Grandoni, Fabrizio I-114, I-490
Greve, Mark I-605
Grigorieff, Serge II-151
Grossi, Roberto I-678
Guerraoui, Rachid II-115
Gupta, Anupam I-262, I-312, I-690
Guruswami, Venkatesan I-360, I-617

Haase, Christoph II-575
Habermehl, Peter II-466
Hähnle, Nicolai I-299
Hajiaghayi, MohammadTaghi I-67
Halldórsson, Bjarni V. I-641
Halldórsson, Magnús M. I-641
Hanusse, Nicolas I-19
Harks, Tobias I-79
Hirschkoff, Daniel II-454
Hofmann, Martin II-199
Husfeldt, Thore I-426, I-727

Iacono, John I-164
Immorlica, Nicole I-67
Ishai, Yuval I-152
Ito, Tsuyoshi I-140

Jacobs, Tobias I-527
Jain, Kamal II-273
Jančar, Petr II-478
Jerrum, Mark I-396
Jiménez, Rosa M. I-238
Jørgensen, Allan Grønlund I-605

Karbyshev, Aleksandr II-199
Kaski, Petteri I-727
Kazeykina, Anna II-273
Khot, Subhash I-250, I-617
Kiefer, Stefan II-539
Kieroński, Emanuel II-357
Kissinger, Aleks II-297
Kleinberg, Robert II-100
Klimm, Max I-79
Köbler, Johannes I-384
Koivisto, Mikko I-727
Král', Daniel I-55
Kratsch, Stefan I-653
Krishnaswamy, Ravishankar I-312, I-324
Krovi, Hari I-540
Kuhnert, Sebastian I-384
Kuperberg, Denis II-563
Kupferman, Orna II-76
Kushilevitz, Eyal I-152, I-451
Kučera, Antonín II-478

Laber, Eduardo I-527
Labourel, Arnaud II-502
Laekhanukit, Bundit I-176
Laird, James II-187
Lampis, Michael II-369
Lanese, Ivan II-442
Laubner, Bastian I-384
Leal, Raul II-381
Lee, Troy I-475
Leonardi, Stefano I-114
Levin, Asaf I-336
Levy, Avivit I-43
Libert, Benoît I-127
Ligett, Katrina II-430
Li, Jian I-188
Litow, Bruce I-420
Lombardy, Sylvain II-563
Losievskaja, Elena I-641
Lucier, Brendan I-90
Lu, Pinyan I-275
Luttenberger, Michael II-539

Magniez, Frédéric I-540
Mahini, Hamid I-67
Makarychev, Konstantin I-594
Maneva, Elitza I-102
Manokaran, Rajsekar I-594
Marcinkowski, Jerzy II-357
Martínez, Conrado I-238
McIver, Annabelle II-223
Medina, Moti II-139
Meinicke, Larissa II-223
Mertzios, George B. II-333
Meyer, Roland II-466
Michaliszyn, Jakub II-357
Mitsou, Valia II-369
Mitzenmacher, Michael I-213
Molinaro, Marco I-527
Monien, Burkhard I-1
Monmege, Benjamin II-587
Montanari, Andrea I-213
Montanari, Angelo II-345
Monti, Angelo II-490

Morgan, Carroll II-223
Mubayi, Dhruv I-438
Muscholl, Anca II-52

Nagarajan, Viswanath I-262, I-324, I-690
Nanongkai, Danupon I-176
Naor, Joseph (Seffi) I-287, II-273
Niemeier, Martin I-299

O'Donnell, Ryan I-617
Orlandi, Alessio I-678
Ostrovsky, Rafail II-236, II-249
Ouaknine, Joël II-22, II-575
Oualhadj, Youssouf II-527
Özkan, Özgür I-164
Ozols, Maris I-540

Pagh, Rasmus I-213
Panagiotou, Konstantinos I-348
Parys, Paweł I-515
Pătraşcu, Mihai I-715
Patt-Shamir, Boaz II-261
Peleg, David II-261
Perdrix, Simon II-285
Pérez, Jorge A. II-442
Perković, Ljubomir I-19
Pin, Jean-Éric II-151
Popat, Preyas I-617
Porat, Ely I-43
Pous, Damien II-454
Pruhs, Kirk I-312
Puppis, Gabriele II-345
Pyrga, Evangelia II-430

Ramachandran, Vijaya I-226
Raman, Rajeev I-678
Ranzato, Francesco II-211
Ravi, R. I-262, I-690
Rensink, Arend II-309
Rink, Michael I-213
Roland, Jérémie I-540
Rosenberg, Adin II-76
Rothvoß, Thomas I-490
Rubinfeld, Ronitt I-565
Rudra, Atri I-629
Rué, Juanjo I-372

Saket, Rishi I-360
Sala, Pietro II-345
Sangiorgi, Davide II-442
Sankowski, Piotr I-114
Sau, Ignasi I-372, II-333
Schmitt, Alan II-442
Schnoebelen, Philippe II-64
Segev, Danny I-582
Seidl, Helmut II-199
Shalom, Mordechai II-333
Shapira, Natalie I-43
Shi, Elaine II-405
Shpilka, Amir I-408
Silvestri, Riccardo II-490
Skutella, Martin I-299
Song, Dawn II-405
Sviridenko, Maxim I-594
Szegedy, Mario I-641

Thilikos, Dimitrios M. I-372
Thorup, Mikkel I-715
Truelsen, Jakob I-605
Tscheuschner, Tobias I-1
Tulsiani, Madhur I-617
Turán, György I-438

Uurtamo, Steve I-629

Vallentin, Frank I-31
van Stee, Rob I-336
Venema, Yde II-381
Verbitsky, Oleg I-384
Verschae, José I-299
Volkovich, Ilya I-408

Wahlén, Martin I-426
Wahlström, Magnus I-653
Walukiewicz, Igor II-52
Wattenhofer, Roger II-38
Weinreb, Enav I-451
Welzl, Emo I-18
Wiese, Andreas I-299
Williams, Ryan II-393
Wimmel, Harro II-466
Woodruff, David P. I-463
Worrell, James II-22, II-575
Wu, Yi I-617

Xia, Mingji I-666
Xie, Ning I-565

Yao, Andrew C. I-702
Yi, Ke I-188
Yung, Moti I-127, I-702

Zadimoghaddam, Morteza II-115
Zaks, Shmuel II-333
Zeitoun, Marc II-587
Zetzsche, Georg II-175
Zhang, Qin I-188
Zhang, Shengyu I-475
Zhao, Yunlei I-702