A Fast Network-Decomposition Algorithm and Its Applications to Constant-Time Distributed Computation[*]

(Extended Abstract)

Leonid Barenboim[1,**], Michael Elkin[2,***], and Cyril Gavoille[3]

[1] Open University of Israel, Israel
leonidb@openu.ac.il
[2] Ben-Gurion University of the Negev, Israel
elkinm@cs.bgu.ac.il
[3] LaBRI - Universite de Bordeaux, Bordeaux, France
gavoille@labri.fr

Abstract. A partition $(C_1, C_2, ..., C_q)$ of $G = (V, E)$ into clusters of strong (respectively, weak) diameter d, such that the supergraph obtained by contracting each C_i is ℓ-colorable is called a strong (resp., weak) (d, ℓ)-network-decomposition. Network-decompositions were introduced in a seminal paper by Awerbuch, Goldberg, Luby and Plotkin in 1989. Awerbuch et al. showed that strong $(exp\{O(\sqrt{\log n \log\log n})\}, exp\{O(\sqrt{\log n \log\log n})\})$-network-decompositions can be computed in distributed deterministic time $exp\{O(\sqrt{\log n \log\log n})\}$. Even more importantly, they demonstrated that network-decompositions can be used for a great variety of applications in the message-passing model of distributed computing. Much more recently Barenboim (2012) devised a distributed randomized constant-time algorithm for computing strong network decompositions with $d = O(1)$. However, the parameter ℓ in his result is $O(n^{1/2+\epsilon})$.

In this paper we drastically improve the result of Barenboim and devise a distributed randomized constant-time algorithm for computing strong $(O(1), O(n^{\epsilon}))$-network-decompositions. As a corollary we derive a constant-time randomized $O(n^{\epsilon})$-approximation algorithm for the distributed minimum coloring problem. This improves the best previously-known $O(n^{1/2+\epsilon})$ approximation guarantee. We also derive other improved distributed algorithms for a variety of problems.

Most notably, for the extremely well-studied distributed minimum dominating set problem currently there is no known deterministic poly-

[*] A full version of this paper with all proofs omitted from the current version due to lack of space is available online [10].

[**] Part of this work has been performed while the author was a postdoctoral fellow at a joint program of the Simons Institute at UC Berkeley and I-Core at Weizmann Institute.

[***] This research has been supported by the Israeli Academy of Science, grant 593/11, and by the Binational Science Foundation, grant 2008390.

© Springer International Publishing Switzerland 2015
C. Scheideler (Ed.): SIROCCO 2015, LNCS 9439, pp. 209–223, 2015.
DOI: 10.1007/978-3-319-25258-2_15

logarithmic -time algorithm. We devise a *deterministic* polylogarithmic-time approximation algorithm for this problem, addressing an open problem of Lenzen and Wattenhofer (2010).

1 Introduction

1.1 Network-Decompositions

In the distributed message-passing model a communication network is represented by an n-vertex graph $G = (V, E)$. The vertices of the graph host processors that communicate over the edges. Each vertex has a unique identity number (ID) consisting of $O(\log n)$ bits. We consider a synchronous setting: computation proceeds in rounds, and each message sent over an edge arrives by the beginning of the next round. The running time of an algorithm is the number of rounds from the beginning until all vertices terminate. Local computation is free.

A *strong* (respectively, *weak*) *diameter* of a cluster $C \subseteq V$ is the maximum distance $\text{dist}_{G(C)}(u, v)$ (resp., $\text{dist}_G(u, v)$) between a pair of vertices $u, v \in C$, measured in the induced subgraph $G(C)$ of C (resp., in G). A partition $(C_1, C_2, ..., C_q)$ of $G = (V, E)$ into clusters of strong (resp., weak) diameter d, such that the supergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \{C_1, C_2, ..., C_q\}$, $\mathcal{E} = \{(C_i, C_j) \mid C_i, C_j \in \mathcal{V}, i \neq j, \exists v_i \in C_i, v_j \in C_j, (v_i, v_j) \in E\}$ obtained by contracting each C_i is ℓ-colorable is called a *strong* (resp., *weak*) (d, ℓ)*-network-decomposition.*

Network-decompositions were introduced in a seminal paper by Awerbuch et al. [3]. The authors of this paper showed that strong $(exp\{O(\sqrt{\log n \log\log n})\}$, $exp\{O(\sqrt{\log n \log\log n})\})$-network-decompositions can be computed in deterministic distributed $exp\{O(\sqrt{\log\log\log n})\}$ time. Even more importantly they demonstrated that many pivotal problems in the distributed message passing model can be efficiently solved if one can efficiently compute (d, ℓ)-network-decompositions with sufficiently small parameters. In particular, this is the case for Maximal Independent Set, Maximal Matching, and $(\Delta + 1)$-Vertex-Coloring.

The result of [3] was improved a few years later by Panconesi and Srinivasan [46] who devised a deterministic algorithm for computing strong $(exp\{O(\sqrt{\log n})\}$, $exp\{O(\sqrt{\log n})\})$-network-decompositions in $exp\{O(\sqrt{\log n})\}$ time. Awerbuch et al. [1] devised a deterministic algorithm for computing strong $(O(\log n), O(\log n))$-network-decomposition in time $exp\{O(\sqrt{\log n})\}$. Around the same time Linial and Saks [40] devised a randomized algorithm for weak $(O(\log n), O(\log n))$-network-decompositions with $O(\log^2 n)$ time. More generally, the algorithm of Linial and Saks [40] can compute weak $(\lambda, O(n^{1/\lambda} \log n))$-network-decompositions or weak $(O(n^{1/\lambda}), \lambda)$-network-decompositions in time $O(\lambda \cdot n^{1/\lambda} \log n)$.

Observe, however, that all these algorithms [3,46,40] require super-logarithmic time, for all choices of parameters. In ICALP'12 the first-named author of the current paper [5] devised a randomized algorithm for computing strong $(O(1), n^{1/2+\epsilon})$-network-decomposition in $O(1/\epsilon)$ time. Unlike the algorithms of [3,46,40], the algorithm of [5] requires *constant* time. Its drawback however is its very high parameter $\ell = n^{1/2+\epsilon}$. In the current paper we alleviate this drawback, and devise a randomized algorithm for computing strong $(exp\{O(\lambda)\}, n^{1/\lambda})$-network-decomposition in time $exp\{O(\lambda)\}$. In other words, the parameter λ of our new decompositions can be made n^{ϵ}, for an arbitrarily small constant $\epsilon > 0$, while the running time is still *constant* (specifically, $exp\{O(1/\epsilon)\}$).

1.2 Constant-Time Distributed Algorithms

In their seminal paper titled "What can be computed locally?" [44] Naor and Stockmeyer posed the following question: which distributed tasks can be solved in *constant* time? This question is appealing both from theoretical and practical perspectives. From the latter viewpoint it is justified by the emergence of huge networks. The number of vertices in the latter networks may be so large that even mildest dependence of the running time on n may make the algorithm prohibitively slow.

Naor and Stockmeyer themselves [44] showed that certain types of weak colorings can be computed in constant time. A major breakthrough in the study of distributed constant time algorithms was achieved though a decade after the paper of [44] by Kuhn and Wattenhofer [35]. Specifically, Kuhn and Wattenhofer [35] showed that an $O(\sqrt{k}\Delta^{1/\sqrt{k}} \log \Delta)$-approximate minimum dominating set[1] can be computed in $O(k)$ randomized time. Here $\Delta = \Delta(G)$ is the maximum degree of the input graph G, and k is a positive possibly constant parameter.

An approximation algorithm for another fundamental optimization problem, specifically, for the *minimum coloring* problem, was devised by Barenboim [5] as an application of his aforementioned algorithm for computing network-decompositions. Specifically, it is shown in [5] that an $O(n^{1/2+\epsilon})$-approximation for the minimum coloring problem can be computed in $O(1/\epsilon)$ randomized time. (In the minimum coloring problem one wishes to color the vertices of the graph properly with as few colors as possible.) Observe that since approximating the minimum coloring problem up to a factor of $n^{1-\epsilon}$ is NP-hard [28,25,50], the algorithm of [5] inevitably has to employ very heavy local computations.

In the current paper we employ our improved network-decomposition procedure to come up with a significantly improved constant-time approximation algorithm for the minimum coloring problem. Specifically, our randomized algorithm provides an $O(n^{\epsilon})$-approximation for the minimum coloring problem in $exp\{O(1/\epsilon)\}$ time, for an arbitrarily small constant $\epsilon > 0$. We also devise a randomized $O(n^{\epsilon})$-approximation algorithm for the *minimum t-spanner* problem with running time $exp\{O(1/\epsilon)\} + O(t)$, for any arbitarily small constant $\epsilon > 0$. (A subgraph $G' = (V, H)$ of a graph $G = (V, E)$, $H \subseteq E$, is a *t-spanner* of G if for every $u, v \in V$, $\text{dist}_{G'}(u, v) \leq t \cdot \text{dist}_G(u, v)$. In the *minimum t-spanner* problem the objective is to compute a t-spanner of the input graph G with as few edges as possible.)

Ajtai et al. [2] demonstrated that triangle-free n-vertex graphs admit an $O(\sqrt{n}/\sqrt{\log n})$-coloring. Kim [30] showed that this existential bound is tight. We devise a randomized $O(n^{1/2+\epsilon})$-coloring algorithm for triangle-free graphs

[1] A subset $U \subseteq V$ in a graph $G = (V, E)$ is a *dominating set* if for every $v \in V \setminus U$ there exists $u \in U$, such that $(u, v) \in E$. In the *minimum dominating set* (henceforth, MDS) problem the goal is to find a minimum-cardinality dominating set of G.

with running time $O(1/\epsilon)$. More generally, we devise a randomized $O(n^{1/k+\epsilon})$-coloring algorithm for graphs of girth greater than $g = 2k, k \geq 2$, with running time $O(1/\epsilon^2)$. Both results apply for any arbitrarily small $\epsilon > 0$, and, in particular, they show that such graph can be colored with a reasonably small number of colors in constant time. Together with our drastically improved constant-time approximation algorithm for the minimum coloring problem, these results significantly expand the set of distributed problems solvable in constant time.

Most our algorithms for constructing network-decompositions use only short messages (i.e., messages of size $O(\log n)$ bits), and employ only polynomially-bounded local computations. Although in general graphs our algorithms for $O(n^{1/\epsilon})$-approximate minimum coloring require large messages, our $O(n^{1/2+\epsilon})$-coloring and $O(n^{1/k+\epsilon})$-coloring algorithms for triangle-free graphs and graphs of large girth employ short messages. Hence the latter coloring algorithms are suitable to serve as building blocks for various tasks. Despite that the number of colors is superconstant, in many tasks it does not affect the overall running time, so the entire task can be performed very quickly. For example, if the colors are used for frequency assignment or code assignment tasks, the running time will not be affected by the number of colors. Instead, the range of frequencies or codes will be affected. However, this is unavoidable in the worst case, in view of the lower bounds on the chromatic number of triangle free graphs and graph of large girth.

1.3 The Minimum Dominating Set Problem

The MDS problem is one of the most fundamental classical problems of distributed graph algorithms. Jia et al. [29] devised the first efficient randomized $O(\log \Delta)$-approximation algorithm for the MDS problem with running time $O(\log n \log \Delta)$. Their result was improved and generalized by Kuhn and Wattenhofer [35] who devised a randomized $O(\sqrt{k}\Delta^{1/\sqrt{k}} \log \Delta)$-approximation algorithm for the problem with time $O(k)$.

The results of [29,35] spectacularly advanced our understanding of the distributed complexity of the MDS problem. However, both these algorithms [29,35] are randomized, and no efficient deterministic distributed algorithms with a non-trivial approximation guarantee for general graphs are currently known. Lenzen and Wattenhofer [38] devised such algorithms for graphs with bounded arboricity. Below we provide a quote from their paper:
"To the best of our knowledge, the deterministic distributed complexity of MDS approximation on general graphs is more or less a blind spot, as so far neither fast (polylogarithmic time) algorithms nor stronger lower bounds are known".

In this paper we address this blind spot and devise a deterministic $O(n^{1/k})$-approximation algorithm for the MDS problem with time $O((\log n)^{k-1})$. Similarly to our approximation algorithms for the minimum coloring and the minimum t-spanner problems, this algorithm is also a consequence of our algorithms for constructing network-decompositions. However, for the MDS we use a deterministic version of these algorithms, while for the minimum coloring and minimum t-spanner problems we use a randomized version. Also, we present a variant of our MDS approximation algorithm that employs only polynomially-bounded local computations, requires $O((\log n)^{k-1})$ time, and provides an $O(n^{1/k}\log \Delta)$ approximation.

1.4 Additional Results

We also use our algorithms for computing network-decompositions for devising algorithms for computing *low-intersecting partitions.* Low-intersecting partitions were introduced by Busch et al. [16] in a paper on universal Steiner trees. A *low-intersecting* (α,β,γ)*-partition* $\mathcal{P}$ of a graph G is the partition of the vertex set V such that: (1) Every cluster C in $\mathcal{P}$ has strong diameter at most $\alpha \cdot \gamma$.
(2) For every vertex $v \in V$, a ball $B_\gamma(v)$ of radius γ around v intersects at most β clusters of $\mathcal{P}$.

Busch et al. showed that given a hierarchy of low-intersecting partitions with certain properties (see [16] for details) one can construct a universal Steiner tree. (See [16] for the definition of universal Steiner tree.) Also, vice versa, given universal Steiner tree they showed that one can construct a low-intersecting partition. They constructed a low-intersecting partition with $\alpha = 4^k, \beta = k\cdot n^{1/k}$, and arbitrary γ.

We devise a distributed randomized algorithm that constructs low-intersecting $((O(\gamma)^k, n^{1/k}, \gamma)$-partitions in time $(O(\gamma))^k \log^{2/3} n$ in general graphs and in $(O(\gamma))^k \cdot exp\{O(\sqrt{\log\log n})\}$ time in graphs of girth $g \geq 6$. This algorithm employs only short messages and polynomially-bounded local computations.

Comparing this result with the algorithm of Busch et al. [16] we note that the partition of [16] has smaller radius. (It is $\gamma \cdot (O(1))^k$ instead of $(O(\gamma))^k$ in our case.) On the other hand, the intersection parameter β of our partitions is smaller. (It is $n^{1/k}$ instead of $k \cdot n^{1/k}$.) In particular, the intersection parameter in the construction of [16] is always $\Omega(\log n)$, while ours can be as small as one wishes. Finally, and perhaps most importantly, the algorithm of [16] is not distributed, and seems inherently sequential.

1.5 Comparison of Our and Previous Techniques

Basically, our algorithms for computing network-decompositions can be viewed as a randomized variant of the deterministic algorithm of Awerbuch et al. [3]. The algorithm of Awerbuch et al. [3] computes iteratively ruling sets for subsets of high-degree vertices in a number of supergraphs. These supergraphs are induced by certain graph partitions which are computed during the algorithm. (A subset $U \subseteq V$ of vertices is called an (α,β)-ruling set if any two distinct vertices $u,u' \in U$ are at distance at least α one from another, and every $v \in V \setminus U$ not in a ruling set has a "ruler" $u \in U$ at distance at most β from v.) As a result of the computation the algorithm of [3] constructs a partition into clusters of diameter at most α, such that the supergraph induced by this partition has arboricity at most β. The algorithm of [3] then colors this partition with $O(\beta)$ colors in time $O(\beta \log n)\cdot O(\alpha)$. (The running time of the algorithm is $O(\beta \log n)$ when running on an ordinary graph. The running time is multiplied by a factor of $O(\alpha)$, because the coloring algorithm is simulated on a supergraph whose vertices are clusters of diameter $O(\alpha)$.) The fact that the running time in the result of [3] is (roughly speaking) the product $\alpha \cdot \beta$ of the parameters of the resulting network-decomposition is the reason that Awerbuch et al [3] made an effort to balance these parameters, and set both of them to be equal to $exp\{O(\sqrt{\log n \log\log n})\}$. The algorithm of Panconesi and Srinivasan [46] is closely related to that of [3] except that it invokes a sophisticated doubly-recursive scheme for computing ruling sets via network-decompositions, and vice versa. This ingenious idea enables [46] to balance the parameters and running time better. Specifically, they are all equal to $2^{O(\sqrt{\log n})}$.

Our algorithm is different from [3,46] in two respects. First, we replace a quite slow (it requires $O(\log n)$ time) deterministic procedure for computing ruling sets by a constant-time randomized one. Note that *generally* computing $(O(1), O(1))$-ruling sets requires $\Omega(\log^* n)$ time [39], but we only need to compute them for *high-degree vertices* of certain supergraphs. This can be easily done in randomized constant time. Second, instead of coloring the resulting partition with $O(\beta)$ colors in $O(\beta \log n) \cdot O(\alpha)$ time, we color it in $O(\beta \cdot n^\epsilon)$ colors in $O(1/\epsilon) \cdot O(\alpha)$ time by a simple randomized procedure, or in $O(\beta^2 \log^{(t)} n)$ colors in $O(t) \cdot O(\alpha)$ time, for a parameter $t > 0$, by a deterministic algorithm Arb-Linial [6]. Hence the number of colors is somewhat greater than in [3,46], but the running time is constant.

The algorithm of Linial and Saks [40] is inherently different from both [3,46] and from our algorithm. It runs for $O(\log n)$ phases, each of which constructs a collection of clusters of diameter $O(\log n)$ at pairwise distance at least 2 which covers at least half of all remaining vertices. The running time of the algorithm of [40], similarly to [3] and [46], is the product of the number of phases and clusters' diameter. Hence the approach of [40] appears to be inherently incapable to give rise to a constant time algorithm.

Our deterministic variant of the network-decomposition procedure is the basis for our deterministic approximation algorithm for MDS. Our deterministic variant is closer to the algorithm of [3] than our randomized one. The main difference between our deterministic variant and the algorithm of [3] is that we use a different much faster coloring procedure for the supergraph induced by the ultimate partition.

1.6 Related Work

Network-decompositions for general graphs were studied in [1,4]. Dubhashi et al. [20] used network decompositions for constructing low-stretch dominating sets. Recently, Kutten et al. [36] extended Linial-Saks network-decompositions to hypergraphs. Many authors [26,34,49] studied network-decompositions for graphs with bounded growth. Distributed approximation algorithms is a vivid research area. See, e.g., [43] and the references therein. Distributed graph coloring is also a very active research area. See a recent monograph [9], and the references therein. Schneider et al. [48] devised a distributed coloring algorithm whose performance depends on the chromatic number of the input graph. However, the algorithm of [48] provides no non-trivial approximation guarantee. Efficient distributed algorithms for constructing sparse undirected spanners can be found in [21,18]. Baswana and Sen [12] devised an approximation algorithm for the minimum t-spanner problem that computes a solution with $O(tn^{1+2/(t+1)})$ expected edges in $O(t^2)$ rounds. For centralized approximation algorithms for the minimum t-spanner problem, see [31,23,12,13].

2 Preliminaries

For a subset $V' \subseteq V$, the graph $G(V')$ denotes the subgraph of G induced by V'. The *degree* of a vertex v in a graph $G = (V, E)$, denoted $\deg_G(v)$, is the number of edges incident on v. A vertex u such that $(u, v) \in E$ is called a *neighbor* of v in G. The *neighborhood* of v in G, denoted $\Gamma_G(v)$, is the set of neighbors of v in G. If the graph G can be understood from context, then we omit the underscript $_G$. For a vertex $v \in V$, the set $v \cup \Gamma(V)$ is denoted by $\Gamma^+(v)$. For a set $W \subseteq V$, we denote by $\Gamma^+(W)$ the set $W \cup \bigcup_{w \in W} \Gamma(w)$. The *distance* between a pair of vertices $u, v \in V$, denoted $\mathrm{dist}_G(u, v)$, is the length of the shortest path between u and v in G. The *diameter* of G is the maximum distance between a pair of vertices in G. The *chromatic number* $\chi(G)$ of a graph G is the minimum number of colors that can be used in a proper coloring of the vertices of G.

3 Network Decomposition

3.1 Procedure Decompose

In this section we devise an algorithm for computing an $(O(1), O(n^\epsilon))$-network-decomposition in $O(1)$ rounds, for an arbitrarily small constant $\epsilon > 0$. More generally, our algorithm computes a $(3^k, O(k \cdot n^{2/k} \cdot \log^2 n))$-network-decomposition Q in $O(3^k \cdot \log^* n)$ rounds, for any positive parameter $k, 1 \leq k \leq \log n$, along with an $O(k \cdot n^{2/k} \cdot \log^2 n)$-coloring φ of the supergraph induced by Q. (The $\log^* n$ term can be eliminated from the running time at the expense of increasing the number of colors used by φ by a multiplicative factor of $\log^{(t)} n$, for an arbitrarily large constant t. We will later show that the multiplicative factor of k in the second parameter of the network decomposition can also be eliminated without affecting other parameters.) The algorithm is called *Procedure Decompose*. The procedure runs on some supergraph $\hat{G} = (\hat{V}, \hat{E})$ of the original graph G. Each vertex $C \in \hat{V}$ is a cluster (i.e., a subset of vertices) of the original graph $G = (V, E)$, and different clusters are disjoint. Observe that generally it may happen that $V \neq \cup_{C \in \hat{V}} C$. The procedure accepts as input the supergraph $\hat{G}$, the number of vertices n of G, the parameter k, and an upper bound s on the number of vertices of the supergraph $\hat{G}$. It also accepts as input two numerical parameters ϵ and t. The parameter $\epsilon > 0$ is a sufficiently small positive constant and $t > 0$ is a sufficiently large integer constant. Initially the supergraph is G itself, with each vertex v forming a singleton cluster $\{v\}$. Hence initially it holds that $n = s$. The procedure is invoked recursively. After each invocation the current supergraph $\hat{G}$ is replaced with a supergraph on fewer vertices, and s is updated accordingly. The parameter n, however, remains unchanged throughout the entire execution.) As a result of an execution of Procedure Decompose every vertex v in $\hat{G}$ is assigned a label $label(v)$. The value of $label(v)$ is equal to the color $\varphi(C_v)$ of the cluster C_v of Q which contains v.

Procedure Decompose partitions the graph $\hat{G}$ into two vertex-disjoint subgraphs with certain helpful properties. Specifically, one of the subgraphs has a sufficiently small maximum degree that allows us to compute a network decomposition in it directly and efficiently. The other subgraph can be partitioned into a sufficiently small number of clusters with bounded diameter. The latter property is used to construct a supergraph whose vertices are formed from the clusters. Since the number of clusters is sufficiently small, the number of vertices of the supergraph is small as well. Then our algorithm proceeds recursively to compute a network decomposition of the new supergraph, using fresh labels that have not been used yet. The recursion continues for k levels. Then each vertex is assigned the label of the supernode it belongs to. (Supernodes of distinct recursion levels may be nested one inside the other. In this case an inner supernode receives the label of an outer supernode. A vertex of the original graph G receives the (same) label of all supernodes it belongs to. Notice that a vertex belongs to exactly one supernode in each recursion level.) This completes the description of the algorithm. Its pseudocode is provided below. (See Algorithm 1.)

The algorithm employs two auxiliary procedures that are described in detail in the full version of this paper [10]. The procedures succeed with high probability, i.e., with probability $1 - 1/n^c$, for an arbitrarily large constant c. The first procedure is called *Procedure Dec-Small*. It accepts a graph G with at most n vertices and maximum degree at most d. Procedure Dec-Small accepts also as input two numerical parameters, ϵ and t, which are relayed to it from Procedure Decompose. Recall that $\epsilon > 0$ is a sufficiently small constant and t is a sufficiently large integer constant. The procedure computes an $O(\min\{d \cdot n^{\epsilon}, d^2\})$-coloring of G in $O(\log^* n)$ time. (The time is $O(1)$ if $d > n^{\epsilon}$. Another variant of this procedure computes an $O(d^2 \log^{(t)} n)$-coloring in $O(t)$ time, for an arbitrarily large positive integer t.) Observe that for any integer $p > 0$, a proper p-coloring of a graph G is also a $(0, p)$-network-decomposition of G. (There are p labels, and each cluster consists of a single vertex. Thus the diameter of the decomposition is 0.) Procedure Dec-Small returns a $(0, p)$-network-decomposition S on line 5. It also returns a labeling function $label_S$ for vertices of a subset A. (We will soon describe how this subset is obtained.) The labeling $label_S$ also serves as a proper coloring for the supergraph induced by S.

The second procedure which is invoked by our algorithm is called *Procedure Partition.* This randomized procedure accepts as input an s-vertex supergraph $\hat{G} = (\hat{V}, \hat{E})$ and a parameter $q < \frac{|\hat{V}|}{2c \cdot \log n}$, and partitions $\hat{V}$ into two subsets A and B, such that $\hat{G}(A)$ and $\hat{G}(B)$ have the following properties. The subgraph $\hat{G}(A)$ has maximum degree $O(q \log n)$. The subgraph $\hat{G}(B)$ consists of $O(|V|/q) = O(s/q)$ clusters of diameter at most 2 with respect to $\hat{G}$. The procedure contracts each such cluster into a supernode. Let $\mathcal{B}$ denote the resulting set of supernodes and $\mathcal{G}(\mathcal{B}) = (\mathcal{B}, \mathcal{E}(\mathcal{B}))$ the resulting supergraph. Specifically, the vertex set of $\mathcal{G}(\mathcal{B})$ is $\mathcal{B}$, and its edge set is $\mathcal{E}(\mathcal{B}) = \{(C, C') \mid C, C' \in \mathcal{B},\ \exists u \in C, u' \in C',$ such that $(u, u') \in \hat{E}\}$. Procedure Partition returns the subset $A \subseteq \hat{V}$ and the set of supernodes $\mathcal{B}$.

The clusters in B are obtained by computing a dominating set D of B of size $O(|V|/q)$. Each vertex in D becomes a leader of a distinct cluster. Each vertex in $B \setminus D$ selects an arbitrary neighbor in D and joins the cluster of this neighbor. Consequently, in all clusters all vertices are at distance at most 1 from the leader of their cluster. Hence all clusters have diameter at most 2. Initially, each vertex of V joins the set D with probability $1/q$. Then the set B is formed by the vertices of D and their neighbors. Finally, the set A is formed by the remaining vertices, i.e., $A = V \setminus B$. In this stage the procedure returns the set of nodes A and the set of supernodes $\mathcal{B}$ which is obtained from B, and terminates. This completes the description of Procedure Partition.

Algorithm 1. Procedure Decompose($\hat{G}, n, k, s, \epsilon, t$)

```
1: /* $c$ is an arbitrarily large positive constant */
2: **if** $s \le 2c \cdot n^{1/k} \log n$ **then**
3:    return Dec-Small($\hat{G}, n, s, \epsilon, t$)
      /* Compute directly a $(0, O(s^2))$-network-decomposition of $\hat{G}$. */
4: **else**
5:    $(A, \mathcal{B}) :=$ Partition($\hat{G}, q := n^{1/k}$)
      /* Partition $\hat{G}$ into $A$ and $\mathcal{B}$. The maximum degree of $\hat{G}(A)$ is $O(n^{1/k} \log n)$.*/
6:    $(S, label_S) :=$ Dec-Small($\hat{G}(A), n, n^{1/k} \log n, \epsilon, t$)
      /* Compute directly a $(0, O(n^{2/k} \cdot \log^2 n))$-network-decomposition of $\hat{G}(A)$. */
7:    $(L, label_L) :=$ Decompose($\mathcal{G}(\mathcal{B}), n, k, \frac{s}{n^{1/k}}$)
      /* A recursive invocation on the supergraph $\mathcal{G}(\mathcal{B})$ that contains at most $\frac{s}{n^{1/k}}$
      supernodes. */
8:    **for** each vertex $v$ of $\hat{G}$, **in parallel, do**
9:       **if** $v \in S$ **then**
10:         $label(v) := label_S(v)$
11:      **else if** $v \in L$ **then**
12:         $label(v) := label_L(v) + \Lambda$
            /* $\Lambda = \gamma \cdot \lfloor n^{2/k} \cdot \log^2 n \rfloor$, where $\gamma$ is a sufficiently large constant to be
            determined later. */
13:      **end if**
         /* The labeling function $label$ on $S \cup L$ is defined by: for a cluster $C \in S$
         (respectively, $C \in L$) it applies to it the function $label_S()$ (resp., $label_L() + \Lambda$).
         */
14:   **end for**
15:   return $(S \cup L, label)$
16: **end if**
```

The recursive invocation of Procedure Decompose on line 7 returns a network decomposition L for the supergraph $\mathcal{G}(B)$. The for-loop (lines 8-14) adds (in parallel) $\Lambda = \gamma \cdot \lfloor n^{2/k} \log^2 n \rfloor$ to the color of each cluster of the network decomposition L_0 of $\mathcal{G}(B)$, where γ is a sufficiently large constant to be determined later. Since the number of colors used in each recursive level is at most Λ, this loop guarantees that colors used for clusters created on different recursion levels are different. This is because the labeling returned by procedure Dec-Small on line 6 for clusters of S employs the palette $[\Lambda]$ while the labeling computed in lines 11 - 13 for clusters of L employs labels which are greater than Λ. The termination condition of the procedure is the case $s = O(n^{1/k} \log n)$, i.e., when the number s of vertices in the supergraph $\hat{G}$ is already small. At this point the maximum degree of $\hat{G}$ is small as well (at most $s-1$), and so coloring the supergraph (by Procedure Dec-Small) results in a sufficiently good network decomposition.

Observe that our main algorithm will invoke the procedure on the original graph G. Hence in the first level of the recursion $\hat{G} = G$, and each supernode is actually a node of G. In the second recursion level it is executed on the supernodes of nodes of the original graph G. In the third level it is executed on supernodes of supernodes, etc. Consequently, starting from the second recursion level supernodes have to be simulated using original nodes of the network. To this end each cluster that forms a supernode selects a leader which is used for simulating the supernodes. Moreover, the leader is used to simulate all nested supernodes to which it belongs. Our supernodes are obtained by at most k levels of nesting. In each level of nesting a supernode is a cluster of diameter at most 2 in a graph whose nodes are lower-level supernodes. Hence a simulation of a single round on such a supergraph will require up to 3^{k+1} rounds.

Next we provide several lemmas that will be used for the analysis of the algorithm. We leave the parameters ϵ and t unspecified in all lemmas in this section, because they have no effect on the analysis.

Lemma 31. *Suppose that all invocations of auxiliary procedures of Procedure Decompose have succeeded. Then the invocation computes a* $(3^{k-1}-1, O(k \cdot n^{2/k} \cdot \log^2 n))$*-network-decomposition.*

Recall that the auxiliary procedures Dec-Small and Partition succeed with probability $1 - 1/n^c$, for an arbitrarily large constant c. Each of these procedures is invoked at most $k \leq \log n$ times during the execution of Procedure Decompose. Therefore, the probability that all executions of Procedure Dec-Small and Procedure Partition succeed is at least $(1 - 1/n^c)^{2 \log n} \approx 1 - \frac{1}{n^c/2 \log n}$. Since c is an arbitrarily large constant, all executions of the auxiliary procedures succeed, with high probability. Hence Procedure Decompose computes a $(3^k, O(k \cdot n^{2/k} \cdot \log^2 n))$-network-decomposition, with high probability.

The next lemma analyzes the running time of the algorithm.

Lemma 32. *Let* $T_{part}(n, q)$ *(respectively,* $T_{dec}(n, d)$*) denote the running time of Procedure Partition invoked with parameters* n *and* q *(resp., Procedure Dec-Small invoked with parameters* n *and* d*). We will assume that both these running times are monotone non-decreasing in both parameters. Then the running time of Procedure Decompose is* $O(3^k \cdot (T_{part}(n, n^{1/k}) + T_{dec}(n, 2c \cdot n^{1/k} \log n)))$.

Procedure Dec-Small and Procedure Partition are provided and analyzed in the full version of this paper[10]. Next we state the main results obtained by plugging these procedures into Procedure Decompose. See [10] for the proofs.

Theorem 33. *For any parameter $k, 1 \leq k \leq \log n$, Procedure Decompose computes a $(3^k, O(k \cdot n^{2/k} \cdot \log^2 n))$-network-decomposition along with the corresponding $O(k \cdot n^{2/k} \cdot \log^2 n)$-labeling function in time $O(3^k \cdot \log^* n)$, with high probability. Alternatively, one can also have the second parameter equal to $O(k \cdot n^{2/k} \log n)$ and the running time $O(3^k \cdot k)$.*

It follows that an $(O(1), n^\delta)$-network-decomposition of an arbitrary n-vertex graph along with a proper n^δ-labeling for it can be computed by a randomized algorithm, in $O(1)$ time, with high probability. Additional variants of the algorithm can be found in [10].

4 Applications

We use our network-decomposition techniques to obtain improved algorithms for a variety of problems. The full description of all these applications appear [10]. Due to lack of space we provide here just a few notable results.

4.1 An Approximation Algorithm for the Coloring Problem

The results described in the previous sections (Theroem 33) imply an approximation algorithm for the optimization variant of the coloring problem. A distributed approximation algorithm for the graph coloring problem (based on an $(O(1), O(n^{1/2+\epsilon}))$-network decomposition) was given in [5]. We describe here a generalization of that algorithm which works with any network-decomposition. The generalized algorithm starts by computing a $(3^k - 1, O(n^{1/k} \log n))$-network-decomposition Q with an $O(n^{2/k} \log n)$-labeling $label(\cdot)$ for it. Then in each cluster C the entire induced subgraph $G(C)$ is collected into the leader vertex v_C of C. The leader vertex v_C computes locally the optimum coloring φ_C for C. Finally, v_C broadcasts (a table representation of φ_C) to all vertices of C. Each vertex u that receives this broadcast computes its final color $\psi(u)$ by $\psi(u) = \langle \varphi_C(u), label(u) \rangle$. The running time of this algorithm is the sum of the time required to compute the decomposition Q (i.e., $O(3^k \cdot k^2)$) with the time required for the computation of the colorings φ_C. The latter is dominated by the diameter of Q, times a small constant. The overall running time is therefore $O(3^k \cdot k^2)$. The result is summarized below.

Theorem 41. *For any n-vertex graph $G = (V, E)$ and an integer parameter $k = 1, 2, ...$, an $O(n^{2/k} \log n)$-approximation of the optimal coloring for G can be computed in $O(3^k \cdot k^2)$ time.*

In particular, by setting the parameter k to be an arbitrarily large constant we get a distributed $O(n^\epsilon)$-approximation algorithm for the coloring problem with a *constant* running time, for an arbitrarily small constant $\epsilon > 0$. (The running time is $O(3^{\lceil 1/\epsilon \rceil} \cdot \frac{1}{\epsilon^2})$.) This greatly improves the current state-of-the-art constant-time distributed approximation algorithm for the coloring problem due to [5], which provides an approximation guarantee of $O(n^{1/2+\epsilon})$.

Note that the algorithm in Theorem 41 requires very heavy (exponential in n) local computations and large messages. The heavy computations are inevitable, because unless $NP = P$, the coloring problem cannot be approximated (in polynomial time) up to a ratio of $n^{1-\epsilon}$, for any constant $\epsilon > 0$ [28,25,50]. On the other hand, in triangle-free graphs we can obtain an algorithm with short messages and polynomially-bounded local computation. See [10].

Theorem 42. *An $O(n^{1/2+\epsilon})$-coloring of triangle-free n-vertex graph can be computed in $O(1/\epsilon)$ distributed randomized time, using short messages and polynomially-bounded local computations.*

4.2 An Approximation Algorithm for the Minimum Dominating Set Problem

In this section we employ our network-decomposition algorithm in order to derive approximation algorithms for the minimum dominating set problem. We need the following notion. For positive integer parameters α, β, σ, an (α, β)-network-decomposition Q of a graph $G = (V, E)$ is called *σ-separated* if the clusters of Q can be β-colored in such a way that every pair of clusters $C, C' \in Q$ which are colored by the same color are at distance at least σ from one another, i.e., $\text{dist}_G(C, C') \geq \sigma$. Observe that an ordinary network decomposition is 2-separated.

Suppose that we are given a 3-separated (d, ℓ)-network-decomposition Q of a graph G. For each cluster $C \in Q$, we compute in parallel a dominating set $D \subseteq \Gamma^+(C)$ of C, such that D has minimum cardinality among all dominating sets $D' \subseteq \Gamma^+(C)$ of C. The computation of D is performed by collecting the topology of the clusters and their neighborhoods by the leaders of respective clusters, performing the computation locally using exhaustive search[2], and broadcasting the results to the vertices of the clusters and their neighbors. Since the weak diameter of the clusters is at most d, this requires $O(d)$ rounds. The next lemma show that the resulting set obtained by taking the union of the dominating sets in all clusters constitutes an ℓ-approximate minimum dominating set of the input graph G.

Lemma 43. *For a 3-separated (d, ℓ)-network-decomposition Q, suppose that we have computed a minimum dominating set $D_C \subseteq \Gamma^+(C)$ of C, for each cluster $C \in Q$. Then $|\bigcup\{D_C \mid C \in Q\}| \leq \ell \cdot |MDS(G)|$.*

In the full version of this paper [10] we devise a routine that computes a strong $((O(\log n))^{k-1}, n^{1/k})$-network-decomposition in deterministic time $(O(\log n))^{k-1}$, for any $k = 1, 2, ...$ Using this network-decomposition in conjunction with Lemma 43 we obtain the following theorem.

Theorem 44. *For an n-vertex graph G, and a positive integer parameter k an $O(n^{1/k})$-approximation for the minimum dominating set problem can be computed in deterministic time $(O(\log n))^{k-1}$.*

[2] One can employ polynomial-time local computations instead of exhaustive search in the expense of increasing the approximation ratio by a factor of $O(\log \Delta)$. See the discussion following Theorem 44 .

To avoid heavy local computations by leaders of clusters, we can run a centralized $O(\log \Delta)$-approximation algorithm for the MDS problem in each cluster. (More precisely, since we need a dominating set for C which can use vertices of $\Gamma^+(C)$, we in fact obtain an instance of the Set Cover problem. This instance has left and right degrees bounded by $\Delta + 1$, and thus one can compute an $O(\log \Delta)$-approximate set cover for this instance in centralized polynomial time. As a result the approximation ratio becomes $O(n^{1/k} \log \Delta)$, while the time stays $(O(\log n))^{k-1}$.

In the full version of this paper[10] we employ our network-decompositions for coloring triangle-free graphs. We also show how our network-decomposition algorithm can be employed to obtain low-intersecting partitions. The latter partitions were used in [16] to construct universal Steiner trees. Finally, we devise a distributed approximation algorithm for the minimum t-spanner problem.

Acknowledgments. The authors are grateful to David Peleg for fruitful discussions that helped obtain some of the results in this paper.

References

1. Awerbuch, B., Berger, B., Cowen, L., Peleg, D.: Fast Distributed Network Decompositions and Covers. J. of Parallel and Distr. Computing 39(2), 105–114 (1996)
2. Ajtai, M., Komlos, J., Szemeredi, E.: A note on Ramsey numbers. Journal of Combinatorial Theory, Series A 29, 354–360 (1980)
3. Awerbuch, B., Goldberg, A.V., Luby, M., Plotkin, S.: Network decomposition and locality in distributed computation. In: Proc. of the 30th Annual Symposium on Foundations of Computer Science, pp. 364–369 (1989)
4. Awerbuch, B., Peleg, D.: Sparse partitions. In: Proc. of the 31st IEEE Symp. on Foundations of Computer Science, pp. 503–513 (1990)
5. Barenboim, L.: On the locality of some NP-complete problems. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part II. LNCS, vol. 7392, pp. 403–415. Springer, Heidelberg (2012)
6. Barenboim, L., Elkin, M.: Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. In: Proc. of the 27th ACM Symp. on Principles of Distributed Computing, pp. 25–34 (2008)
7. Barenboim, L., Elkin, M.: Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. In: Proc. of the 41st ACM Symp. on Theory of Computing, pp. 111–120 (2009)
8. Barenboim, L., Elkin, M.: Deterministic distributed vertex coloring in polylogarithmic time. In: Proc. 29th ACM Symp. on Principles of Distributed Computing, pp. 410–419 (2010)
9. Barenboim, L., Elkin, M.: Distributed Graph Coloring: Fundamentals and Recent Developments. Morgan-Claypool Synthesis Lectures on Distributed Computing Theory (2013)
10. Barenboim, L., Elkin, M., Gavoille, C.: A Fast Network-Decomposition Algorithm and its Applications to Constant-Time Distributed Computation, `http://arxiv.org/abs/1505.05697`
11. Barenboim, L., Elkin, M., Pettie, S., Schneider, J.: The locality of distributed symmetry breaking. In: Proc. of the 53rd Annual Symposium on Foundations of Computer Science, pp. 321–330 (2012)

12. Baswana, S., Sen, S.: A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. Random Structures and Algorithms 30(4), 532–563 (2007)
13. Berman, P., Bhattacharyya, A., Makarychev, K., Raskhodnikova, S., Yaroslavtsev, G.: Improved approximation for the directed spanner problem. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 1–12. Springer, Heidelberg (2011)
14. Bisht, T., Kothapalli, K., Pemmaraju, S.: Super-fast t-ruling sets (Brief Announcement). In: Proc. of the 33th ACM Symposium on Principles of Distributed Computing, pp. 379–381 (2014)
15. Bollobas, B.: Extremal Graph Theory. Dover Publications (2004)
16. Busch, C., Dutta, C., Radhakrishnan, J., Rajaraman, R., Srinivasagopalan, S.: Split and join: strong partitions and universal Steiner trees for graphs. In: Proc. of 53rd Annual IEEE Symp. on Foundations of Computer Science, pp. 81–90 (2012)
17. Cole, R., Vishkin, U.: Deterministic coin tossing with applications to optimal parallel list ranking. Information and Control 70(1), 32–53 (1986)
18. Derbel, B., Gavoille, C., Peleg, D., Viennot, L.: On the locality of distributed sparse spanner construction. In: Proc. of the 27th ACM Symp. on Principles of Distributed Computing, pp. 273–282 (2008)
19. Dinitz, M., Krauthgamer, R.: Directed spanners via flow-based linear programs. In: Proc. of the 43rd ACM Symp. on Theory of Computing, pp. 323–332 (2011)
20. Dubhashi, D., Mei, A., Panconesi, A., Radhakrishnan, J., Srinivasan, A.: Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. Journal of Computer and System Sciences 71(4), 467–479 (2005)
21. Elkin, M.: A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. In: Proc. of the 26th ACM Symp. on Principles of Distributed Computing, pp. 185–194 (2007)
22. Elkin, M., Peleg, D.: The client-server 2-spanner problem with applications to network design. In: Proc. of the 8th International Colloquium on Structural Information and Communication Complexity, pp. 117–132 (2001)
23. Elkin, M., Peleg, D.: Approximating k-spanner problems for $k \geq 2$. Theoretical Computer Science 337(1-3), 249–277 (2005)
24. Erdős, P., Frankl, P., Füredi, Z.: Families of finite sets in which no set is covered by the union of r others. Israel Journal of Mathematics 51, 79–89 (1985)
25. Feige, U., Kilian, J.: Zero Knowledge and the chromatic number. Journal of Computer and System Sciences 57, 187–199 (1998)
26. Gfeller, B., Vicari, E.: A randomized distributed algorithm for the maximal independent set problem in growth-bounded graphs. In: Proc. of the 26th ACM Symp. on Principles of Distributed Computing, pp. 53–60 (2007)
27. Goldberg, A., Plotkin, S., Shannon, G.: Parallel symmetry-breaking in sparse graphs. SIAM Journal on Discrete Mathematics 1(4), 434–446 (1988)
28. Hastad, J.: Clique is Hard to Approximate Within $n^{1-\epsilon}$. In: Proc. of the 37th Annual Symposium on Foundations of Computer Science, pp. 627–636 (1996)
29. Jia, L., Rajaraman, R., Suel, R.: An efficient distributed algorithm for constructing small dominating sets. In: Proc. of the 20th ACM Symp. on Principles of Distributed Computing, pp. 33–42 (2001)
30. Kim, J.H.: The Ramsey number $R(3,t)$ has order of magnitude $t^2/\log t$. Random Structures and Algorithms 7, 173–207 (1995)
31. Kortsarz, G., Peleg, D.: Generating sparse 2-spanners. Journal of Algorithms 17(2), 222–236 (1994)

32. Kothapalli, K., Pemmaraju, S.: Super-fast 3-ruling sets. In: Proc. of the 32nd IARCS International Conference on Foundations of Software Technology and Theoretical Computer Science, pp. 136–147 (2012)
33. Kuhn, F.: Weak graph colorings: distributed algorithms and applications. In: Proc. of the 21st ACM Symposium on Parallel Algorithms and Architectures, pp. 138–144 (2009)
34. Kuhn, F., Moscibroda, T., Wattenhofer, R.: On the locality of bounded growth. In: Proc. of the 24th ACM Symp. on Principles of Distributed Computing, pp. 60–68 (2005)
35. Kuhn, F., Wattenhofer, R.: Constant-time distributed dominating set approximation. Distributed Computing 17(4), 303–310 (2005)
36. Kutten, S., Nanongkai, D., Pandurangan, G., Robinson, P.: Distributed symmetry breaking in hypergraphs. In: Proc. of the 28th International Symposium on Distributed Computing, pp. 469–483 (2014)
37. Lenzen, C., Oswald, Y., Wattenhofer, R.: What can be approximated locally? case study: dominating sets in planar graphs. In: Proc 20th ACM Symp. on Parallelism in Algorithms and Architectures, pp. 46–54 (2010). See also TIK report number 331, ETH Zurich, 2010
38. Lenzen, C., Wattenhofer, R.: Minimum dominating set approximation in graphs of bounded arboricity. In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 510–524. Springer, Heidelberg (2010)
39. Linial, N.: Locality in distributed graph algorithms. SIAM Journal on Computing 21(1), 193–201 (1992)
40. Linial, N., Saks, M.: Low diameter graph decomposition. Combinatorica 13, 441–454 (1993)
41. Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM Journal on Computing 15, 1036–1053 (1986)
42. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press (2005)
43. Nanongkai, D.: Distributed approximation algorithms for weighted shortest paths. In: Proc. of the 46th ACM Symp. on Theory of Computing, pp. 565–573 (2014)
44. Naor, M., Stockmeyer, L.: What can be computed locally? In: Proc. 25th ACM Symp. on Theory of Computing, pp. 184–193 (1993)
45. Panconesi, A., Rizzi, R.: Some simple distributed algorithms for sparse networks. Distributed Computing 14(2), 97–100 (2001)
46. Panconesi, A., Srinivasan, A.: On the complexity of distributed network decomposition. Journal of Algorithms 20(2), 581–592 (1995)
47. Saket, R., Sviridenko, M.: New and improved bounds for the minimum set cover problem. In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) APPROX/RANDOM 2012. LNCS, vol. 7408, pp. 288–300. Springer, Heidelberg (2012)
48. Schneider, J., Elkin, M., Wattenhofer, R.: Symmetry breaking depending on the chromatic number or the neighborhood growth. Theoretical Computer Science 509, 40–50 (2013)
49. Schneider, J., Wattenhofer, R.: A log-star distributed maximal independent set algorithm for growth bounded graphs. In: Proc. of the 27th ACM Symp. on Principles of Distributed Computing, pp. 35–44 (2008)
50. Zuckerman, D.: Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. Theory of Computing 3(1), 103–128 (2007)
51. `http://www.disco.ethz.ch/lectures/ss04/distcomp/lecture/chapter12.pdf`