Laboratoire Bordelais de Recherche en Informatique
UMR 5800 - Université Bordeaux 1, 351, cours de la Libération,
33405 Talence CEDEX, France

Research Report RR-1374-05

On Space-Stretch Trade-Offs for Compact Routing Schemes

by

Ittai Abraham,
Cyril Gavoille, and
Dahlia Malkhi

November, 2005

Contents

On Space-Stretch Trade-Offs for Compact Routing Schemes

Ittai Abraham* Cyril Gavoille† Dahlia Malkhi‡

Abstract

One of the fundamental trade-offs in compact routing schemes is between the *space* used to store the routing table on each node and the *stretch* factor of the routing scheme – the maximum ratio over all pairs between the cost of the route induced by the scheme and the cost of a minimum cost path between the same pair. We study upper and lower bounds for the space-stretch trade-off for distributed routing schemes. Our upper bound results are based on a decomposition into sparse and dense neighborhoods. This decomposition allows to design routing schemes in which storage and header sizes are independent of the aspect ratio of the network. Given an undirected network with arbitrary weights and n arbitrary node names, for any integer $k \geq 1$ we present a routing scheme with asymptotically optimal space-stretch trade-off that does *not* require edge weights to be polynomially bounded. The scheme uses $\widetilde{O}(n^{1/k})$ space routing table at each node, and routes along paths of linear stretch $O(k)$. Using a distributed Kolmogorov Complexity argument, we give a lower bound for the name-independent model that applies even to single-source schemes and does not require a girth conjecture. For any integer $k \geq 1$ we prove that any routing scheme for networks with arbitrary weights and arbitrary node names (even a single-source routing scheme) with stretch strictly less than $2k+1$ requires $\Omega((n \log n)^{1/k})$-bit routing tables.

*School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel. ittaia@cs.huji.ac.il

†Laboratoire Bordelais de Recherche en Informatique, University of Bordeaux, Bordeaux, France. Supported by the project "PairAPair" of the ACI Masses de Données. gavoille@labri.fr

‡School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel, and Microsoft Research, Silicon Valley Center. dalia@microsoft.com

1 Introduction

One of the most basic functionalities of any distributed network is the ability to route messages between pairs of nodes. Given that each node has an arbitrary network identifier, a routing scheme allows any source node to route messages to any destination node, given the destination's network identifier. It is natural to consider a weighted network in which the cost of routing a message is proportional to the cost of the path taken from source to destination. In such a model it is desirable to minimize routing costs by routing on short paths. In this sense the efficiency of a routing scheme is measured by its *stretch factor*, the maximum ratio over all source destination pairs, between the cost of routing from the source to the destination and the cost of a minimum cost path. The trivial solution to routing on shortest paths with stretch factor 1 is for each node to store a routing table with $(n-1)$ entries that contains the next hop of an all pairs shortest path algorithm. This solution is very expensive as it requires each node to store $\Omega(n \log n)$ bits. Thus, network designers are faced with two conflicting goals: reduce both the stretch factor and the size of the routing tables.

For a weak variant of this fundamental problem, called *labeled routing*, both lower bounds and asymptotically optimal upper bounds are known (see [28]). In this version of the problem, the designer of a solution may pick node names that contain (bounded size) information about their location in the network. This variant is useful in many aspects of network theory, but less so in practice: Knowledge of the labels needs to be disseminated to all potential senders, as these labels are not the addresses by which nodes of an *existing* network, e.g., an IP network, are known. Furthermore, if the network may admit new joining nodes, all the labels may need to be re-computed and distributed to any potential sender. Finally, various recent applications pose constraints on nodes addresses that cannot be satisfied by existing labeled-routing schemes. E.g., Distributed Hash Tables (DHTs) require nodes names in the range $[1..n]$, or ones that form a binary prefix.

In this paper we assume a network with arbitrary node names and arbitrary edge weights. This model is called the *name-independent* model because the designer of the routing scheme has no control over node names. This routing problem may appear daunting: In order to route to a node, we must first somehow gain knowledge about its location in the network, but we must do so without exceeding the distance to the target.

The two, long-standing (1989, Awerbuch et al. [5]) core challenges in compact routing theory are:

1. Find a routing scheme with the lowest worst case space-stretch trade-off.

2. Prove that its space-stretch trade-off is tight.

A fundamental difficulty in all previous solution schemes is their heavy dependence on Δ, the network's aspect ratio. When Δ is exponential in n, all previous solutions are not compact. This suggests that there might exist a lower bound associated with the parameter Δ. However, the best known lower bound for name-independent routing [28] does not contain such dependence on Δ, and furthermore, relies on an unproven conjecture. Thus, none of the two fundamental questions above have been answered in a satisfying manner as of yet.

Our contributions. For the upper bound we construct for any $k \geq 1$ a routing scheme with linear stretch factor of $O(k)$ and with $\widetilde{O}(n^{1/k})$-bit routing tables per node and $\widetilde{O}(1)$-bit headers[1].

Theorem 1. *For each weighted n-node graph, and integer $k \geq 1$, there is a polynomial time constructible name-independent routing scheme with stretch factor $O(k)$ that uses $O(k^2 n^{1/k} \log^5 n)$-bit routing tables per node.*

Our scheme is based on a new decomposition into dense and sparse neighborhoods. It uses a subtle combination of combinatorial and geometric routing techniques depending on the density or sparsity of each neighborhood. This decomposition allows us to remove any dependence on the aspect ratio. Our technique bears some similarities to the "measured decent" approach of Krauthgamer et al. [21] in that scales are chosen as a function of a node's local density. However, using a decent based approach for routing fails since it relies only on density changes. Hence searching on a ball with $\Omega(\log n)$ density changes may incur $O(\log n)$ scratch. Our decomposition circumvents this by decomposing both by density change and by diameter change.

Our sparse/dense decomposition technique has interest in its own, and can serve to remove the aspect ratio parameter in many other constructions.

For the lower bound we prove that for any integer $k \geq 1$, any routing scheme for networks with arbitrary weights and arbitrary node names (even a single-source routing scheme) with $\Omega((n \log n)^{1/k})$ space routing tables must have networks in which the stretch is at least $2k+1$. This proves asymptotic tightness of our upper bound.

Theorem 2. *For each integer $k \geq 1$ there is an n-node star with edge weights 1 or k for which every single-source name-independent routing scheme with less than $\frac{1}{9}(n \log n)^{1/k}$ bits of memory requirements has stretch factor at least $2k+1$.*

Our lower bound is obtained using a novel distributed Kolmogorov Complexity argument. Previous lower bounds for routing schemes with stretch of at least $2k+1$ either obtain a trade-off with a forth-root less memory, $\Omega(n^{1/(4k+4)})$ by Peleg and Upfal [27]. Or, as in Thorup and Zwick [28] obtain $\Omega(n^{1/k})$ but their proof relies on an unproven Erdös girth Conjecture [13]. Moreover, our lower bound has a subtle $(\log n)^{1/k}$ factor that was not obtained by [28]. Furthermore, all previous lower bounds apply only for all-pair routing schemes. In contrast, our lower bound uses a local argument that applies even to single-source name-independent routing schemes. Therefore it also proves that Laing's stretch $2k-1$ single-source scheme [22] is optimal up to polylogarithmic storage factors.

[1]The notation $\widetilde{O}(\cdot)$ indicates complexity similar to $O(\cdot)$ up to poly-logarithmic factors.

Related work. The space-stretch trade-off has been extensively studied under various models and extensions. We refer the reader to Peleg's book [26] and to the surveys of Gavoille and Peleg [16, 18] for comprehensive background.

Peleg and Upfal [27] were the first to study this trade-off in a parameterized manner. They give the first lower bound proving that routing schemes for n-node networks with stretch $k \geq 1$ must use a total of $\Omega(n^{1+1/(2k+4)})$ bits of storage. They also provide an upper bound for unweighted networks with unit cost edges. For unit cost networks they achieve $O(k)$ stretch with a total of $O(k^3 n^{1+1/k} \log n)$ bits for all routing tables. For weighted networks, Awerbuch et al. [5, 6] present a routing scheme with exponential stretch of $O(k^2 9^k)$ that requires $\widetilde{O}(n^{1/k})$ bits per node. Arias et al. [4] improve the stretch to $O(k^2 2^k)$ with the same memory bound.

In addition they study schemes with $\widetilde{O}(n^{1/2})$-bit routing tables and obtain stretch 5. Abraham et al. [2] improve the stretch factor to 3 with the same memory bound of $\widetilde{O}(n^{1/2})$. By the lower bound of Gavoille and Gengler [17] any routing scheme with $o(n)$ memory must have stretch of at least 3.

Awerbuch and Peleg use *sparse covers* [7] in order to build a hierarchal routing scheme [8]. Their scheme is based on tree covers with geometrically increasing radii. Therefore there is an inherent geometric factor in their memory requirement. They achieve stretch $O(k^2)$ with $\widetilde{O}(n^{1/k} \log \Delta)$-bit routing tables, and $O(\log \Delta)$ headers, where Δ is the maximum weighted distance between any two nodes divided by the minimum weighted distance between any two unique nodes. Abraham, Gavoille, and Malkhi [1] improve the stretch factor to $O(k)$ with the same memory requirement. This solution is adequate if the network weights are polynomial in the number of nodes. However for arbitrary networks the diameter may be arbitrarily large, for instance $\Delta = \Omega(2^n)$, and solutions based on the geometry of the network may become unusable.

A weaker variant of compact routing is based on the *labeled routing* model. Instead of assuming nodes have arbitrary names, in this model, the network designer is allowed to name the nodes in a topology dependent manner. This paradigm does not provide for a realistic network design, however, the tools devised for its solution have proven useful as building blocks of full routing schemes.

Eilam et al. [12] present a stretch 5 labeled scheme with $\widetilde{O}(n^{1/2})$ memory, whereas Cowen [11] presents a stretch 3 labeled scheme with $\widetilde{O}(n^{2/3})$ memory. Later, Thorup and Zwick [28] improve to stretch 3 using $\widetilde{O}(n^{1/2})$ bits. These three schemes uses $O(\log n)$-bit node names. Thorup and Zwick also give in [28] a generalization of their scheme and using techniques from their distance oracles [29], achieve labeled schemes with stretch $4k-5$ (and even $2k-1$ with handshaking) using $\widetilde{O}(n^{1/k})$-bit routing tables and $o(k \log^2 n)$-bit node names.

Labeled routing on a trees is explored in [14, 28], achieving stretch 1 with $O(\log^2 n / \log\log n)$ bits for local tables and for headers, and this is tight [15]. Laing [22] presents a routing scheme on trees with arbitrary names that obtains stretch $2^k - 1$ with $\widetilde{O}(n^{1/k})$-bit routing tables. With the same bit complexity the author gives a *single-source* routing scheme with stretch $2k-1$. Therefore, compact routing schemes with arbitrary names independent of the aspect ratio are known only for trees, and with a stretch exponentiel in k.

2 Sparse and Dense Neighborhood Decomposition

2.1 Preliminaries

Given is a weighted graph $G=(V,E,\omega)$ of size $n=|V|$ with a non-negative weight function $\omega : E \to \mathbb{R}^+$. Let the cost of a path be the sum of the weights of its edges. For any two nodes $u,v \in V$ let $d(u,v)$ denote the cost of a minimum cost path between u and v. Let Δ denote the *aspect ratio* (normalized diameter) of G, $\Delta = \max_{u\neq v} d(u,v)/\min_{u\neq v} d(u,v)$. In order to avoid dragging a normalization constant, from here on assume that $\min_{u\neq v} d(u,v)=1$. Define the radius r ball around node u, $B(u,r)$, as the set of nodes whose distance is at most r from u, $B(u,r)=\{v \mid d(u,v)\leq r\}$. For any node u, let $T(u)$ denote a minimum cost path spanning tree rooted at u. Given a lexicographic order on the nodes, for any node $u\in V$, set $Z\subseteq V$, and integer $m>0$ define $N(u,m,Z)$ as the m closest nodes from Z to node u, i.e., as the set $N(u,m,Z)=N$ such that $N\subseteq Z$, $|N|=m$ and for all $x\in N$ and $y\in Z\setminus N$ either $d(u,x)<d(u,y)$ or $d(u,x)=d(u,y)$ and x is lexicographically smaller than y. Let I denote the set $I=\{0,1,\ldots,\lceil\log\Delta\rceil\}$. Given a parameter k, let K denote the level set $K=\{0,1,2,\ldots,k\}$. Each node has an arbitrary unique network identifier consisting of polylog(n) bits. Using standard hashing techniques it is possible to generalize the model and assume nodes have arbitrarily long unique labels.

Our solution is based on using a new decomposition into a series of balls around each node that have a combined combinatorial and geometric restriction. Each ball has at least $n^{1/k}$ more nodes than the previous *and* its radius is at least twice the radius of the previous. Formally, for all $u\in V$ and $i\in K$ define the *range* $a(u,i)$ recursively as follows. Define $a(u,0)=0$. Then recursively define $a(u,i+1)$ as the smallest positive integer $j>0$ such that $|B(u,2^j)|\geq n^{1/k}|B(u,2^{a(u,i)})|$ (or let $a(u,i+1)=\log\Delta$ if there does not exist such an integer). For all $u\in V$ and $i\in K$ denote the *neighborhood* ball $A(u,i)$ as the ball who's radius is $2^{a(u,i)}$ around u. Formally, $A(u,i)=\{u\}$ for $i=0$ and $A(u,i)=B(u,2^{a(u,i)})$ for $i>0$.

Intuitively, if the gap between $a(u,i)$ and $a(u,i+1)$ is small, then the neighborhood $A(u,i+1)$ is "dense" relative to the neighborhood $A(u,i)$, otherwise $A(u,i+1)$ is "sparse" relative to $A(u,i)$. A central definition capturing this intuitive notion is the following.

Definition 1 (Dense level). *For $u\in V$ and $i\in K$, define that i is a* dense level *for node u if*

$$a(u,i) \; < \; a(u,i+1) \; \leq \; a(u,i)+3$$

Define that i is a *sparse level* for node u if it is not a dense level. In words, in a dense level, we find at least $n^{1/k}$ times as many nodes as the current level by looking at a ball whose radius is at most 2^3 times the current level.

2.2 Dense Levels

For every $u \in V$ define the *range set of node* u, denoted $L(u)$, as $L(u) = \{a(u,i) \mid i \in K\}$ and define the *extended range set* $R(u)$ as,

$$R(u) = \{i \in I \mid \exists a \in L(u), -1 \leq a - i \leq 4\} \ .$$

Define $F(u,i) = B(u, 2^{a(u,i)-1})$. The main property of dense levels is captured in the following lemma.

Lemma 3 (Dense neighborhoods).
If i is a dense level for u and $v \in F(u,i)$, then $a(u,i) \in R(v)$.

Proof. Recall that $F(u,i) = B(u, 2^{a(u,i)-1})$. Let $v \in F(u,i)$, then $B(v, 2^{a(u,i)-1}) \subseteq A(u,i)$ (see Figure 1), and hence

$$|B(v, 2^{a(u,i)-1})| \ \leq \ |A(u,i)| \ .$$

Since $a(u,i+1) \leq a(u,i) + 3$, then $B(v, 2^{a(u,i)+4}) \supseteq A(u,i+1)$, and hence

$$|B(v, 2^{a(u,i)+4})| \ \geq \ |A(u,i+1)| \ \geq \ |A(u,i)| \cdot n^{1/k} \ .$$

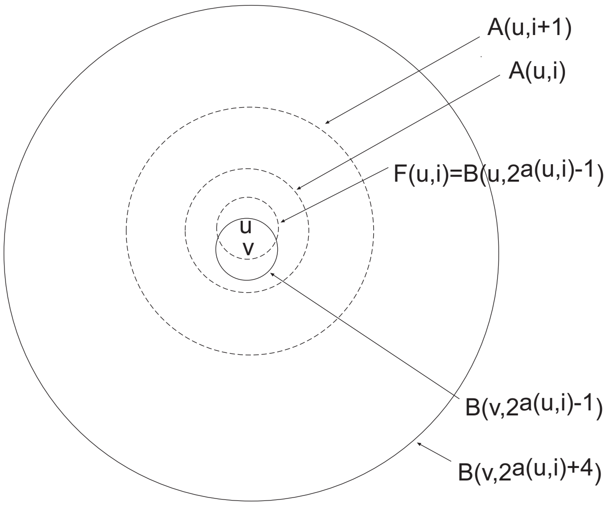

Figure 1: Example of a level i dense neighborhood for node u and a node $v \in F(u,i)$.

Together, these imply $|B(v, 2^{a(u,i)+4})| \geq n^{1/k} \cdot |B(v, 2^{a(u,i)-1})|$. Therefore, there exists some index $a(v,j)$ such that $a(u,i) - 1 \leq a(v,j) \leq a(u,i) + 4$. ☐

2.3 Sparse Levels

We use a low discrepancy cover of 'landmark' nodes. Let $\mathcal{B} = \{B(u, 2^i) \mid u \in V, i \in I\}$, note that $|\mathcal{B}| \leq |V|^2$. We use $k+1$ sets $V = C_0 \supseteq C_1 \supseteq \cdots \supseteq C_k = \varnothing$ of landmarks with the following property.

Property 4. *For all $B \in \mathcal{B}$ and $j \in K$*

$$\left\lfloor \frac{|B|}{n^{j/k} 2 \log n} \right\rfloor \;\leq\; |B \cap C_j| \;\leq\; \left\lceil \frac{|B| 2 \log n}{n^{j/k}} \right\rceil$$

A construction satisfying Property 4 with high probability can be achieved by the following simple randomized sampling procedure[2]. For $i = 1$ to $k-1$ iteratively set C_i to contain each element of C_{i-1} independently, with probability $n^{-1/k}$. If $x \in C_j$, and $x \notin C_{j+1}$, define that node x has *rank* j. For every $u \in V$ and $i \in K$ define the *nearby landmarks* $S(u,i)$ to be the $n^{2/k} \log n$ closest nodes in C_i.

$$S(u,i) = N(u, 4n^{2/k} \log^2 n, C_i)$$

and define $S(u) = \bigcup_{i \in K} S(u,i)$. Define $m(u,i)$ as the highest rank of any node in $A(u,i)$. Formally,

$$m(u,i) = \max \{\ell \in K \mid A(u,i) \cap C_\ell \neq \varnothing\} \;.$$

Define the *center* $c(u,i)$ as the closest node to u from $C_{m(u,i)}$. Let $E(u,i) = B(u, 2^{a(u,i+1)}/6)$. The main property of sparse levels is captured in the following lemma.

Lemma 5 (Sparse neighborhoods).
Let i be a sparse level for u, i.e., $a(u,i+1) > a(u,i)+3$. If $v \in E(u,i)$, then $c(u,i) \in S(v)$.

Proof. Recall that $E(u,i) = B(u, 2^{a(u,i+1)}/6)$ and $m(u,i)$ is the highest rank of any node in $A(u,i)$. Formally,

$$m(u,i) \;=\; \max \{\ell \in K \mid A(u,i) \cap C_\ell \neq \varnothing\} \;.$$

Let $j \in K$ be the index such that $n^{j/k} \log n \leq |A(u,i)| < n^{(j+1)/k} \log n$ then from Property 4 it follows that $m(u,i) \geq j$. For any $v \in E(u,i)$, we have (see Figure 2)

$$c(u,i) \in A(u,i) \subseteq E(u,i) \subseteq B(v, 2^{a(u,i+1)}/3) \subseteq B(u, 2^{a(u,i+1)}/2) \;.$$

Since level i is sparse for u, by definition there are strictly fewer than $n^{1/k}|A(u,i)|$ nodes in $B(u, 2^{a(u,i+1)}/2)$. Therefore

$$|B(v, 2^{a(u,i+1)}/3)| \;<\; n^{1/k}|A(u,i)| \;\leq\; n^{(j+2)/k} \log n \;.$$

From Property 4 and since $m(u,i) \geq j$ it follows that there are less than $4n^{2/k} \log^2 n$ nodes of rank $m(u,i)$ in $B(v, 2^{a(v,i+1)}/3)$. Since $c(u,i) \in B(v, 2^{a(v,i+1)}/3)$ then $c(u,i) \in S(v)$ as required. ∎

[2]This randomized scheme can be derandomized using the method of conditional expectation.

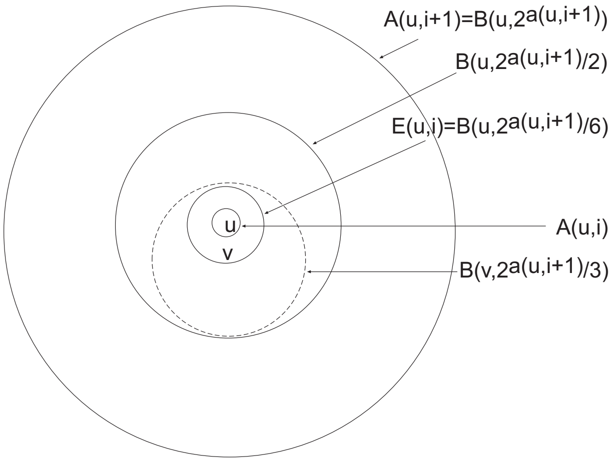

Figure 2: Example of a level i sparse neighborhood for node u and a node $v \in E(u,i)$.

3 A Routing Scheme for Arbitrary Graphs

The goal of this section is to present the construction for the following upper bound:

Theorem 1. *For each weighted n-node graph, and integer $k \geq 1$, there is a polynomial time constructible name-independent routing scheme with stretch factor $O(k)$ that uses $O(k^2 n^{1/k} \log^5 n)$-bit routing tables per node.*

The high level view of the routing scheme is a simple iterative protocol. For phases $i = 1$ to k, search for v as follows: If $A(u,i)$ is sparse, use the sparse neighborhood routing strategy. If $A(u,i)$ is dense, use the dense neighborhood routing strategy. We begin by describing two routing strategies.

3.1 Sparse neighborhood routing strategy

For every center $c(u,i)$ define $T(c(u,i))$ as a minimum cost path tree rooted at $c(u,i)$ that spans all nodes v such that $c(u,i) \in S(v)$. Routing on these trees is done using the following name-independent tree-routing scheme, which is an enhancement of Laing's algorithm [22].

Lemma 6. *For any $k \geq 1$, and for any weighted tree $T = (V, E, \omega)$ and for any designated root $r \in V$, there exists a name-independent error-reporting tree-routing scheme with the following properties:*

1. *Each node stores $O(n^{1/k} \log^3 n)$ bits of routing information.*
2. *For any $j \in K$, the root can perform a j-*bounded search *for destination v. A j-bounded search for v has the following properties: If $v \in N(r, n^{j/k}, V)$ then it reaches v with stretch $2j-1$; Otherwise it returns a negative response to the root incurring a cost of at most $(2j-2)\max\left\{d(r,v) \mid v \in N(r, n^{(j-1)/k}, V)\right\}$*

Proof. Given a tree $T = (V, E)$ with $m \leq n$ nodes and a root $t \in T$. We give each node $v \in V$ three names. Let $a_0, \ldots, a_{m-1}$ denote the nodes of T sorted by increasing distance from the root. Formally, for any two indexes, if $i < j$ then $d_T(r, a_i) < d_T(r, a_j)$ or $d_T(r, a_i) = d_T(r, a_j)$ and a_i is lexicographically smaller than a_j. The first name we give nodes, called their primary name, makes use of words of the alphabet $\Sigma = \{0, 1, 2, \ldots n^{1/k} - 1\}$. Specifically, name $a_0 = r$ as the empty word (ε). Then name $a_1, \ldots, a_{n^{1/k}}$ respectively with one digit in Σ in increasing order $(0), (1), \ldots, (n^{1/k} - 1)$ respectively. Then name each of the next $n^{2/k}$ nodes $a_{1+n^{1/k}}, \ldots, a_{1+n^{1/k}+n^{2/k}}$ by a name in Σ^2 in increasing lexicographic order, $(0, 0), (0, 1), \ldots, (0, n^{1/k} - 1), (1, 0), (1, 1), \ldots, (1, n^{1/k} - 1), \ldots, (n^{1/k} - 1, 0), \ldots, (n^{1/k} - 1, n^{1/k} - 1)$ respectively. Continue this naming process, naming nodes $a_{\sum_{j=0}^{i-1} n^{j/k}}, \ldots, a_{\sum_{j=0}^{i} n^{j/k}}$ respectively by a name in Σ^i until all nodes in T are exhausted, up to at most a k-digit node name in Σ^k. For $0 \leq j \leq k$ let V_j denote the set of nodes whose name contains at most j digits. Next, we give v its name in the labeled tree routing of Thorup and Zwick [28] and Fraigniaud and Gavoille [14]:

Lemma 7. [14, 28] *For every weighted tree T with m nodes there exists a labeled routing scheme that, given any destination label, routes optimally on T from any source to the destination. The storage per node in T, the label size, and the header size are $O(\log^2 m/ \log\log m)$ bits. Given the information of a node and the label of the destination, routing decisions take constant time.*

For a tree T containing a node v, let $\mu(T, v)$ denote the routing information of node v and $\lambda(T, v)$ denote the destination label of v in T as required from Lemma 7. We require from each node v to store $\mu(T, v)$. The second name we assign v is $\lambda(T, v)$. The third node-name makes use of a hash function $h : V \to \Sigma^k$. We require that, for all $0 \leq j \leq k$, $\max_{u\in\Sigma^{j-1}} | \{v \in V_j \mid u \text{ is a prefix of } h(v)\} | \leq 4|\Sigma| \log n = 4n^{1/k} \log n$. This requirement can be fulfilled with high probability using a $O(|\Sigma| \log n)$-wise independent hash function that requires $O(|\Sigma| \log^2 n)$ bits of storage [10, 25]. A node $u \in V$ with name $(x_1, \ldots, x_j)$ stores:

1. Information for labeled tree routing $\mu(T, u)$.
2. A map $v \to \lambda(T, v)$ of all the nodes v whose name is $(x_1, \ldots, x_j, y)$ for all $y \in \Sigma$.
3. The map $v \to \lambda(T, v)$ of the $4n^{1/k} \log n$ closest nodes v (from the root) whose first j indexes of their hash $h(v)$ equal $x_1, \ldots, x_j$. Formally, let $Z = \{z \mid h(z)[1 \ldots j] = x_1, \ldots, x_j\}$ and u stores $v \to \lambda(T, v)$ forall $v \in N(r, 4n^{1/k} \log n, Z)$.

To search from the root r for a node t whose hash is $h(t) = (y_1, \ldots, y_k)$ on a j-bounded search, do the following:

1. *current* $:= r$; *round* $:= 1$;
2. if *round* $= j$ and *current* does not know of t then return to root r with negative response.
3. Otherwise if *current* knows of t's label then route to t.

4. Otherwise route to the node whose name is $(y_1, \ldots, y_{round})$, set $round := round + 1$, and update $current$ to be the current node.
5. goto 2;

Suppose the destination is a node t whose hash is $y_1, \ldots, y_k$ and whose name has length i ($t \in V_i$ and $t \notin V_{i+1}$). The destination will be found after at most i iterations of the algorithm. This is true since if $i = 1$, then the root knows about the destination. Otherwise, at the $i-1$th iteration we reach the node x whose name is $(y_1, \ldots, y_{i-1})$ and due to the properties of the hash function we know that $|\{v \in V_i \mid (y_1, \ldots, y_{i-1}) \text{ is a prefix of } h(v)\}| \leq |\Sigma| \log n$ and hence x stores the label of t.

Since all the nodes that are visited have smaller names than i, then their distance from the root is smaller than $d(r,t)$, so the stretch is bounded by $2i - 1 \leq 2k - 1$.

The storage per node v is $O(|\Sigma| \log^2 n) + O(|\Sigma| \log^2 n) + O(|\Sigma| \log^3 n) = O(n^{1/k} \log^3 n)$, for $\mu(T,v)$, the primary name entries, and the hash name entries. If a j-bounded search is performed and $j < i$ then t may not be found. At the $(j-1)$th iteration, a node whose name is $(y_1, \ldots, y_{j-1})$ will report the error to the root. Since all nodes visited have names with at most $j-1$ digits then the total cost is bounded by $(2j-2)\max_{v \in V_{j-1}}\{d(r,v)\}$. □

For all $u \in V$ and $i \in K$ recall that $E(u,i) = B(u, 2^{a(u,i+1)}/6)$. Given a sparse level i the algorithm routes to the root $c(u,i)$. We prove that $E(u,i) \subseteq T(c(u,i))$ and hence searching from $c(u,i)$ for $v \in E(u,i)$ on the tree $T(c(u,i))$ will succeed with stretch of at most $2k-1$. In order to bound the cost incurred if $v \notin E(u,i)$ every node $u \in V$ stores for every $i \in K$ the index $b(u,i)$. Where $b(u,i)$ is the minimal integer j such that a j-bounded search on $T(c(u,i))$ of any node in $E(u,i)$ succeeds. Since $b(u,i)$ is defined as the minimal index to find in $T(c(u,i))$ all nodes of $E(u,i)$ then we prove that a negative result from this search has cost proportional at most to the diameter of $E(u,i)$. More precisely, we now define the information stored by every node, and the routing algorithm for sparse neighborhoods.

Storage: For any tree T and node $u \in T$, let $\tau(T,u)$ denote the information stored on u that is induced by the name-independent tree-routing scheme of Lemma 6. Every node $u \in V$ stores $\tau(T(v),u)$ for all $v \in S(u)$. In addition, for every $i \in K$, u records $c(u,i)$ and $b(u,i)$.

Routing algorithm for sparse level i:

1. route to the root $c(u,i)$.
2. Perform a $b(u,i)$-bounded search on $T(c(u,i))$ for destination v.
3. If destination is not found, continue to iteration $i+1$.

3.2 Dense neighborhood routing strategy

If i is a dense level, the routing strategy uses tree covers that have a bounded radius. A source of difficulty in arbitrarily weighted graphs is that the logarithm of the diameter may be arbitrarily large ($\gg n$, for example $\Delta = 2^n$). Hence, tree covers of geometrically increasing radii, e.g., as used in [7, 1], require each node to participate in too many partition levels any may require $\Omega(n)$ bits per node. Our solution for general graphs is to let each node maintain routing information only for a limited number of radii that surround its range set. However, special care must be taken to make sure that the destination and all the nodes along the way store information for the same radius that the source is using. Recall that the range set is defined as $L(u) = \{a(u,i) \mid i \in K\}$, and the *extended range set* $R(u)$ is defined as, $R(u) = \{i \in I \mid \exists a \in L(u), -1 \leq a - i \leq 4\}$. For every $i \in I$ define $G_i = (V_i, E_i)$ as the subgraph induced by the nodes $V_i = \{u \mid i \in R(u)\}$. We prove that in a dense level i for u, if v is in $B(u, 2^{a(u,i)-1})$ then $i \in R(v)$ and moreover both source and destination are connected in G_i. Hence a tree cover in G_i may be used for routing. The tree cover is built using the construction of [7] with the improvements of [1].

Lemma 8. **[7, 1]** *For every weighted graph $G = (V, E, \omega)$, $|V| = n$ and integers $k, \rho \geq 1$, there exists a polynomial algorithm that constructs a collection of rooted trees $\mathcal{TC}_{k,\rho}(G)$ such that:*

1. *(Cover) For all $v \in V$, there exists $T \in \mathcal{TC}_{k,\rho}(G)$ such that $B(v, \rho) \subseteq T$.*
2. *(Sparse) For all $v \in V$, $|\{T \in \mathcal{TC}_{k,\rho}(G) \mid v \in T\}| \leq 2kn^{1/k}$.*
3. *(Small radius) For all $T \in \mathcal{TC}_{k,\rho}(G)$, $\mathrm{rad}(T) \leq (2k-1)\rho$, where $\mathrm{rad}(T) = \max_u \{d_T(r,u)\}$.*
4. *(Small edges) For all $T \in \mathcal{TC}_{k,\rho}(G)$, $\mathrm{maxE}(T) \leq 2\rho$, where $\mathrm{maxE}(T) = \max_{e \in E(T)} \{\omega(e)\}$.*

For every $i \in I$ we build a tree cover $\mathcal{TC}_{k,2^i}(G_i)$ only on the graph G_i (note that G_i may have several connected components, so a tree cover is built for each connected component separately). Since $|R(u)| = O(k)$ then every node u participates only in $O(k)$ such tree covers. For all $u \in V$ and $i \in K$, denoting $j = a(u,i)$, define $W(u,i) \in \mathcal{TC}_{k,2^j}(G_j)$ to be the tree such that $B(u, 2^j) \subseteq W(u,i)$. On a dense level i, routing towards a destination begins by routing to the root of the tree $W(u,i)$ and then using a name-independent tree-routing scheme. For all $i \in I$ and $T \in \mathcal{TC}_{k,\rho}(G_i)$ we use the name-independent error-reporting tree-routing scheme of [1].

Lemma 9. **[1]** *For every tree $T = (U, E, \omega)$, $|U| = m$, $U \subset V$, $|V| = n$, and integer k there exists a name-independent tree-routing scheme on T with error-reporting that routes on paths of length bounded by $4\mathrm{rad}(T) + 2k\mathrm{maxE}(T)$, each node requires $O(kn^{1/k}\log^2 n)$ memory bits, and headers are of length $O(\log^2 n)$. Moreover, routing for a non-existent name in T also incurs a (closed) path of length $4\mathrm{rad}(T) + 2k\mathrm{maxE}(T)$ until a negative result is reported back to the source.*

We prove that if i is a dense level for u then $\forall v \in F(u,i)$ we have $i \in R(v)$ (Lemma 3) and hence the tree routing scheme on $W(u,i)$ will reach any node in $F(u,i)$ or report a negative response at a cost proportional to k times the radius of $F(u,i)$ (recall $F(u,i) = B(u, 2^{a(u,i)-1})$). We now define the information stored by every node, and the routing algorithm for dense neighborhoods.

Storage: For any tree T and node $u \in T$, let $\phi(T,u)$ denote the information stored on u that is induced by the name-independent error-reporting tree-routing scheme of Lemma 9. For every $u \in V$ and $i \in R(u)$ node u stores $\phi(T,u)$ for all $T \in \mathcal{TC}_{k,2^i}(G_i)$ such that $u \in T$. In addition, every node $u \in V$ records $w(u,i)$ the root of the tree $W(u,i)$.

Routing algorithm for dense level i:

1. Route to the root $w(u,i)$.
2. Route on the tree $W(u,i)$ to either find destination v or get a negative response.
3. If destination is not found, continue with iteration $i+1$.

3.3 Analysis

Throughout the description below, the source node is denoted u and the destination node is denoted v. Routing from u to v is done by iteratively expanding the search through the neighborhoods of u, $A(u,1), A(u,2), \dots, A(u,k)$ until the destination is found. The routing strategy in each neighborhood $A(u,i)$ depends on whether level i is sparse or dense for node u. If $A(u,i)$ is sparse, we use the sparse neighborhood strategy. If $A(u,i)$ is dense we use the dense neighborhood routing strategy.

The following technical lemmata are used to prove Theorem 1.

Lemma 10. *Let i be a dense level for u, and let $v \in F(u,i)$. Then a dense neighborhood routing strategy from u will reach v.*

Proof. Recall that $R(u) = \{i \mid \exists \ell \in L(u), -1 \le \ell - i \le 4\}$. Denote $a(u,i) = j$, by the properties of dense neighborhood stated in Lemma 3, then for every $x \in F(u,i)$ node x has $a(u,i) \in R(x)$ and hence $x \in G_j$. Recall that the tree $W(u,i) \in \mathcal{TC}_{k,2^j}(G_j)$ is defined to be the tree such that $B_{G_j}(v, 2^j) \subseteq W(u,i)$. Therefore, $F(u,a) \subseteq W(u,i)$ and it is possible to reach v by routing on $W(u,i)$. □

Lemma 11. *Let $i \in K$ be a dense level for $u \in V$. Then the dense neighborhood routing strategy requires $O(k^3 n^{2/k} \log^2 n)$ memory bits and incurs a cost of $O(k \cdot 2^{a(u,i)})$ either to reach v if $v \in B(u, 2^{a(u,i)-1})$ or otherwise to return a negative response to the source u.*

Proof. Storage: For the dense level routing strategy every node maintains $O(k)$ ranges. For each range, every node participates in $O(kn^{1/k})$ trees, each tree requires $O(kn^{1/k} \log^2 n)$ bits. For a total of $O(k^3 n^{2/k} \log^2 n)$ bits.

Routing cost: By combining Lemma 8 and Lemma 9 the cost of searching for v on $G_{a(u,i)}$ is at most $O(k \cdot 2^{a(u,i)})$. From Lemma 10, the destination will actually be found on $G_{a(u,i)}$ if $v \in F(u,i) = B(u, 2^{a(u,i)-1})$. □

Lemma 12. *Let i be a sparse level for u, and let $v \in E(u,i)$. Then the sparse neighborhood routing strategy from u will reach v.*

Proof. Denote $c = c(u,i)$, and recall that c is the closest landmark of highest rank in $A(u,i)$. By the properties of sparse neighborhoods stated in Lemma 5, every node $v \in E(u,i)$ has $c \in S(v)$. Hence, all nodes in $E(u,i)$ store $\tau(T(c))$, including all nodes on the shortest path of $T(c)$ from u to c. Since $b(u,i)$ is defined so that a $b(u,i)$-bounded search on $T(c)$ will find all nodes in $E(u,i)$ then $v \in E(u,i)$ will be found. □

Lemma 13. *Let $i \in K$ be a sparse level for $u \in V$. Then the sparse neighborhood routing strategy requires $O(kn^{3/k} \log^5 n)$ memory bits and incurs a cost of $O(k \cdot (d(u,v) + 2^{a(u,i)}))$ to reach v if $v \in B(u, 2^{a(u,i+1)}/6)$ or otherwise returns a negative response to the source at a cost $O(k \cdot 2^{a(u,i+1)})$.*

Proof. Storage: For the sparse level routing strategy every node maintains k closest landmark sets $S(u,1), \ldots, S(u,k)$. For each set, a node participates in $4n^{2/k} \log^2 n$ trees, each tree requires $O(n^{1/k} \log^3 n)$ bits. For a total of $O(kn^{3/k} \log^5 n)$ bits.

Routing cost: If $v \in E(u,i) = B(u, 2^{a(u,i+1)}/6)$, then by Lemma 12 the routing strategy will route from u to $c(u,i)$ and then to v. By definition $c(u,i) \in A(u,i)$ so the path to $c(u,i)$ incurs a cost of at most $d(u, c(u,i)) \le 2^{a(u,i)}$. Then by Lemma 6 the $b(u,i)$-bounded search will find v at cost at most $(2k-1)d(c(u,i),v) \le (2k-1)(d(u,v) + d(u,c(u,i))) \le (2k-1)(d(u,v)+2^{a(u,i)})$. So the total cost is bounded by $2^{a(u,i)} + (2k-1)(d(u,v)+2^{a(u,i)}) = O(k(d(u,v) + a(u,i)))$ as required. Otherwise if $v \notin E(u,i) = B(u, 2^{a(u,i+1)}/6)$ then the round-trip cost of reaching $c(u,i)$ and returning is at most $2^{a(u,i)+1}$. Then by Lemma 6 the cost of a $b(u,i)$-bounded search is bounded by $(2k-2)2^{a(u,i+1)}/6$. This is true since $b(u,i)$ is defined so that all the $n^{b(u,i)/k}$ closest nodes in $T(c(u,i))$ all belong to $E(u,i)$. Hence the total cost for a negative answer is $2^{a(u,i)+1} + (2k-2)2^{a(u,i+1)} = O(k \cdot 2^{a(u,i+1)})$. □

We now prove the main theorem.

Proof of Theorem 1. For storage, by combining Lemma 13 and Lemma 11 it follows that every node stores $O(k^2 n^{3/k} \log^5 n)$-bit routing tables. For stretch analysis, let $i \in K$ be the first iteration index in which v is found. There are two cases to consider.

1. If level $i-1$ is sparse for u, then given that v is not found in iteration $i-1$, by Lemma 13, v is not inside $E(u,i-1)$, and hence, $d(u,v) \ge 2^{a(u,i)}/6$.
2. Otherwise, level $i-1$ is dense for u, and again, v is not found in iteration $i-1$. In this case, by Lemma 11, v is not inside $F(u,i-1) = B(u, 2^{a(u,i-1)}/2)$. By density, $a(u,i) \le a(u,i-1) + 3$. Putting these two facts together, we have $d(u,v) \ge 2^{a(u,i-1)}/2 \ge 2^{a(u,i)-3}/2$.

In either case, $2^{a(u,i)} = O(d(u,v))$. From Lemma 13 and Lemma 11, reaching v on level i will cost $O(k \cdot 2^{a(u,i)})$ if i is dense or $O(k(d(u,v) + 2^{a(u,i)}))$ if i is sparse. Hence in both cases, level i searches are bounded by $O(k \cdot d(u,v))$. For the cost of the negative responses, note that the highest level that fails is $i-1$. From Lemma 13 and Lemma 11 the cost of a negative response for level j is either $O(k \cdot 2^{a(u,j)})$ for a dense level, or $O(k \cdot 2^{a(u,j+1)})$ for a sparse level. Hence, the total cost of negative response is at most $\sum_{j=0}^{i-1} O(k \cdot 2^{a(u,j+1)}) = O(k \cdot 2^{a(u,i)}) = O(k \cdot d(u,v))$. □

4 A Lower Bound for Single-Source Routing Scheme

The goal of this section is to prove the following lower bound:

Theorem 2. *For each integer $k \geq 1$ there is an n-node star with edge weights 1 or k for which every single-source name-independent routing scheme with less than $\frac{1}{9}(n \log n)^{1/k}$ bits of memory requirements has stretch factor at least $2k+1$.*

We begin with some historical perspective on this bound. First, it is worth mentioning the following result, since it emphasizes a subtle difference between arbitrary and uniform edge-weighted trees. Indeed, we have:

Theorem 14 ([1]). *Every unweighted n-node tree has a name-independent single-source routing scheme with constant stretch factor and with $\widetilde{O}(1)$ memory requirements.*

Our lower bound technique to prove Theorem 2 is new and is interesting in its own right. All previous lower bounds in the field of compact routing [27, 19, 20, 9, 28] are based on structural information of some worst-case graphs. Roughly speaking, the previous technique requires to construct a suitable family of $\Omega(2^m)$ graphs, and then to argue that from any compact routing scheme on any graph of the family, one can identify every edge of the graph, therefore forcing the memory requirement of the scheme to be at least $\Omega(m)$ bits in total, in the worst case graph of the family, and thus $\Omega(m/n)$ for at least one node. Typically, for lower bounds on stretched routing schemes, the family is chosen to contain only graphs of girth at least $2k+2$. And so the route of any routing scheme of stretch $< 2k+1$ between adjacent nodes is forced to traverse a given edge. With this technique, [27] and then [28] have showed that, for each integer $k \geq 1$, any routing scheme has stretch at least $\Theta(k)$ if less than $\Omega(n^{1/k})$ bits of memory requirements is used. The exact lower bound on the stretch is $2k+1$ assuming that the maximum number of edges in n-node graphs of girth at least $2k+2$ is $\Omega(n^{1+1/k})$ (or equivalently the number of such graphs is $2^{\Omega(n^{1+1/k})}$). This is related to an 1963 Erdös Conjecture [13] that has been proven only for $k = 1, 2, 3$ and 5. Currently, the best lower bound on the number of edges that holds for every k is $\Omega(n^{1+1/(3\lfloor k/2 \rfloor - 1)})$ [23]. Although it applies on both labeled and name-independent routing schemes, this technique cannot be used to prove Theorem 2, for several reasons. First, the technique provides a lower bound for the *total* memory requirement. However, the total memory requirement for the single-source name-independent routing problem is no more than $\widetilde{O}(n)$ bits with stretch 1, since the source only needs to store an n-entry translation table between the name and a tree-routing label of a node. And, labeling routing schemes with $\widetilde{O}(1)$-bit labels are known for trees [14, 28]. Secondly, the total amount of structural information available in a tree is $O(n \log n)$ bits. Therefore, no lower bound on the total memory requirement better than this can be established using the standard technique. Finally, the standard technique does not apply to single-source routing schemes.

A contrario, we overcome the need to rely on the Erdös Conjecture using a completely different approach. In essence, all the previous lower bounds were based on static information contained in some graphs. Here we use the fact that during the routing of a message from the source to the destination, the header cannot accumulate information too quickly without violating the memory bound of the nodes, though there are no restrictions on the header length. E.g., the information transported by the header of a message leaving the first node, intuitively, cannot exceed the memory bound of the source. A fine analysis of this dynamic behavior of all the routes allows us to conclude with the bound. Note that we also improve by a $(\log n)^{1/k}$ factor previous lower bounds. No bounds better than $\Omega(n^{1/k})$ are possible with the previous proof techniques since there are less than $2^{\frac{1}{2}n^{1+1/k}}$ graphs of girth $2k+2$ or more. Indeed, a simple argument shows that any graph with at least $\frac{1}{2}n^{1+1/k}$ edges contains a cycle of length at most $2k$ (cf. [3]).

We present below the formal proof that makes an original use of Kolmogorov Complexity. We first define in Section 4.1 precisely the routing scheme model and its memory requirements. Then, in Section 4.2 we provide background on Kolmogorov Complexity, and we describe the counter-example in Section 4.3. Finally, the proof of the final theorem is given in Section 4.4.

4.1 Routing scheme model and memory requirements

We use a general and standard model for routing schemes, originally introduced by Peleg and Upfal in [27]. A *routing scheme* is a function that associates with each graph G:

1. a *name* to each node u of G, denoted by $\mathrm{name}(u)$;
2. a *port number* to each ordered pair (u, v) of adjacent nodes of G, denoted by $\mathrm{port}(u, v)$, such a way the neighbors of u gets distinct port numbers taken from $\{1, \ldots, \deg(u)\}$; and
3. a *routing function* R for G, described below.

Roughly speaking, when a message with header h_i arrives at some node x_i through the input port q_i, then x_i computes the pair $(h_{i+1}, p_{i+1}) = R(x_i, h_i, q_i)$ thanks to R. Then, x_i attaches to the message the new header h_{i+1}, and forwards it through the output port p_{i+1} to get the next node x_{i+1} (see Figure 3).

More formally, a routing function R for a graph G with name and port assignments must satisfy that for every source-destination pair u, v of G there exists a walk $u = x_0, \ldots, x_t = v$ from u to v in G, a sequence $h_0, \ldots, h_t$ of headers, and two sequences $q_0, \ldots, q_t$ and $p_1, \ldots, p_{t+1}$ of ports numbers such that: for every $i \in \{0, \ldots, t-1\}$,

$R(x_i, h_i, q_i) = (h_{i+1}, p_{i+1})$ with $h_0 = \text{name}(v)$, $q_0 = p_{t+1} = 0$, $p_{i+1} = \text{port}(x_i, x_{i+1})$, and $q_{i+1} = \text{port}(x_{i+1}, x_i)$. Observe that the destination is always given at the source by its name. The condition $q_0 = p_{t+1} = 0$ is just a convention. In general, for an edge u, v, we have $\text{port}(u, v) \neq \text{port}(v, u)$. However, input and output ports corresponding to the same incident edge at any node x_i must match, i.e., we have $q_i = p_{i+1}$ if the ports use the same edge in x_i.

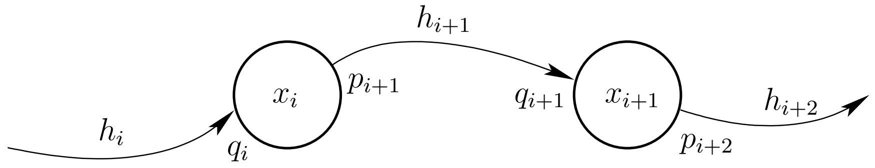

Figure 3: Model of a routing function: $R(x_i, h_i, q_i) = (h_{i+1}, p_{i+1})$.

The restriction of R at node u is denoted by R_u. The *memory requirement* of a routing function R at node u is the length (in bits) of the smallest program, say, written in C, JAVA, LISP, or FORTRAN, that implements the function R_u. This program includes all the constants and the data structures needed to compute R_u.

4.2 Kolmogorov Complexity background

In the lower bound proof we propose it is quite convenient to speak in terms of *Kolmogorov Complexity* to formally quantify the "information" contained in a routing table or in some headers. We give a short background of the required definitions and notations (see [24] for a more comprehensive introduction to this concept).

Given a finite binary string S, we denote by $C(S)$ the *Kolmogorov Complexity of* S, that is the length (in bits) of the smallest program, written in a fixed language[3], that outputs S and halts. By extension, if A is an integer, a sequence of integers, or any object that can be canonically mapped to binary strings (so any element of a given enumerable set), we denote by $C(A)$ the Kolmogorov Complexity of its binary string canonical[4] representative. E.g., the Kolmogorov Complexity of an integer is the Kolmogorov Complexity of its binary representation. By counting the number of distinct programs of length at most a given size, it is clear that any set of cardinality at least n should possess an element S with $C(S) \geq \log_2 n$. More generally we define $C(A \mid B)$ as the Kolmogorov Complexity of A *given* B, that is the size of the smallest program with input B that prints out A and halts. Finally, the Kolmogorov Complexity of a couple (A, B) is denoted by $C(A, B)$, that is the length of a program that prints out both A and B, and a way to tell them apart.

[3] It is easy to show that the language chosen in the above definition affects the value of $C()$ by an additive term only which depends on the language but is independent of S.

[4] Again this complexity depends with only on the canonical definition mapping, a constant additive term independent of an individual object.

For the lower bound we need a technical result about the existence of sequences in which every subset and subsequence has high Kolmogorov Complexity relative to its cardinality. Specifically, let L be a *sequence*, i.e., an ordered list of items. We denote by $L[i]$ the ith item of L and by $|L|$ its length. A *sub-sequence of* L is a sequence that can be obtained from L by removing some elements of L, and the set of distinct items of L is denoted by $L^* = \{L[i] \mid 1 \le i \le |L|\}$.

Lemma 15. *For every n large enough, there is a sequence L of $\lfloor n/2 \rfloor$ distinct integers taken from $\{1, \ldots, n\}$ such that:*

1. *every subset A of L^* has $|A| \le C(A) + 2\log n$; and*
2. *every sub-sequence B of L has $|B| \le (2C(B) + 4\log n)/\log(C(B) + 2\log n)$.*

Proof. We lower and upper bound the Kolmogorov Complexity of L^* expressed as a function of $|A|$ and of $|B|$, and thus derive an upper bound on $|A|$ and $|B|$.

Let $\ell = \lfloor n/2 \rfloor$. The number of sequences composed of ℓ distinct integers taken from $\{1, \ldots, n\}$ is $\ell!\binom{n}{\ell}$. Because all the elements of such sequences are distinct, there are exactly $\ell!$ sequences L having a common given set L^*. Therefore, there is such a sequence L with $C(L \mid n) \ge \log\left(\ell!\binom{n}{\ell}\right)$ and $C(L^* \mid n) \ge \log\binom{n}{\ell}$. Note that $|L| = |L^*| = \ell$. Let us prove the first claim of Lemma 15. Consider any subset $A \subseteq L^*$. We can code L^*, and so upper bound $C(L^* \mid n)$, by merging the sets A and $L^* \setminus A$. In other words,

$$C(L^* \mid n) \;\le\; C(A) + C(L^* \setminus A \mid A) + \log n + O(1)$$

where the term "$\log n$" stands for the coding of the delimiter between the code for A and for $L^* \setminus A$. More precisely, the coding of L^* (given n) is composed of the binary string $S = S_1 S_2 S_3$ where:

1. S_3 is a coding of A, so of length $C(A)$;
2. S_2 is a coding of $L^* \setminus A$ given A; and
3. S_1 is the binary representation of $|S_2| = C(L^* \setminus A \mid A)$ on $\lceil \log 2n \rceil$ bits exactly. Clearly, the number of possible subsets $L^* \setminus A$ is at most 2^n, thus $|S_2| \le n + O(1) \le 2n$ for n large enough.

So given n, one can extract from S the $\lceil \log(2n) \rceil$ first bits to get S_1, and then extract S_2 and S_3. Let $a = |A|$. The set $L^* \setminus A$ is composed of $\ell - a$ integers taken from $\{1, \ldots, n\} \setminus A$. So, once A has been constructed, it remains at most $\binom{n-a}{\ell-a}$ possible candidates for $L^* \setminus A$. In other words, $C(L^* \setminus A \mid A) \le \log\binom{n-a}{\ell-a} + O(1)$. Therefore, we have the upper bound for the Kolmogorov Complexity of L^* that is:

$$C(L^* \mid n) \;\le\; C(A) + \log\binom{n-a}{\ell-a} + \log n + O(1)\ .$$

And, by the previous lower bound, it yields:

$$\log\binom{n}{\ell} \;\le\; \log\binom{n-a}{\ell-a} + C(A) + \log n + O(1)$$

Set $m = C(A) + \log n + O(1)$ so that Ineq. (1) rewrites:

$$\binom{n}{\ell} \;\le\; \binom{n-a}{\ell-a} \cdot 2^m \quad\Leftrightarrow\quad \frac{n!}{\ell!} \;\le\; \frac{(n-a)!}{(\ell-a)!} \cdot 2^m \tag{1}$$

Using the Stirling approximation $u! \sim \left(\frac{u}{e}\right)^u \sqrt{2\pi u}$, we have for all u and v large enough:

$$\frac{u^u}{v^v} \cdot e^{v-u} \;\le\; \frac{u!}{v!} \;\le\; \frac{u^u}{v^v} \cdot e^{v-u}\sqrt{2\pi u}$$

Plugging in Ineq. (1) the later lower and upper bound, we obtain that:

$$\frac{n^n}{\ell^\ell} \;\le\; \frac{(n-a)^{n-a}}{(\ell-a)^{\ell-a}} \cdot 2^m \sqrt{2\pi(n-a)} \tag{2}$$

Let us show that:

Claim 1. *For all integers a, ℓ, n such that $0 \le a \le \ell$ and $0 < \ell \le n$,*

$$\frac{(n-a)^{n-a}}{(\ell-a)^{\ell-a}} \;\le\; \frac{n^{n-a}}{\ell^{\ell-a}}\ .$$

Proof. This inequality is equivalent to

$$\left(\frac{n-a}{n}\right)^{n-a} \;\le\; \left(\frac{\ell-a}{\ell}\right)^{\ell-a} \quad\Leftrightarrow\quad \left(1-\frac{a}{n}\right)^n \;\le\; \left(1-\frac{a}{\ell}\right)^\ell \quad\Leftrightarrow\quad f(n) \;\le\; f(\ell)$$

where $f(x) = (1 - a/x)^x$ with $x \ge a$ and $x \ne 0$. It remains to show that f is decreasing for every $x \ge a$. We compute:

$$f'(x) \;=\; \left(1-\frac{a}{x}\right)^{x-1}\left(\frac{a}{x} + \left(1-\frac{a}{x}\right)\ln\left(1-\frac{a}{x}\right)\right)$$

which has the same sign as $y + (1-y)\ln(1-y)$ where $y = a/x$. The later term is negative as $y \to 0$, so $f'(x) \le 0$ for every $x \ge a$, and so f decreases as claimed. $\square$

Assume now that $2^m\sqrt{2\pi n} < n^a/\ell^a$. By Claim 1, we have therefore that:

$$2^m\sqrt{2\pi n} \cdot \frac{(n-a)^{n-a}}{(\ell-a)^{\ell-a}} \;<\; \frac{n^a}{\ell^a} \cdot \frac{n^{n-a}}{\ell^{\ell-a}} = \frac{n^n}{\ell^\ell}$$

which is a contradiction with Ineq. (2). Therefore, we have $2^m\sqrt{2\pi n} \ge (n/\ell)^a$, that is

$$a \;\le\; \frac{1}{\log(n/\ell)}\left(m + \log\sqrt{2\pi n}\right) \quad\Leftrightarrow\quad |A| \;\le\; \frac{1}{\log(n/\ell)}\left(C(A) + \log n + O(1) + \log\sqrt{2\pi n}\right)$$

For $\ell = \lfloor n/2 \rfloor$, we have $\log(n/\ell) \ge 1$, and for n large enough, we have $\log n + O(1) + \log\sqrt{2\pi n} \le 2\log n$, and therefore:

$$|A| \;\le\; C(A) + 2\log n$$

completing the proof of the first claim.

We now prove the second claim of Lemma 15. Consider any sub-sequence B of L, and let $b = |B|$. Our objective here is to upper bound b. We can code L, and so upper bound $C(L \mid n, \ell)$, by:

1. the sub-sequence B;
2. the sub-sequence $L \setminus B$; and
3. an ℓ-bit long binary string with b ones.

To reconstruct L we read each bit of the binary string and output either an element of B or of $L \setminus B$ depending of whether the bit is set of not. So we have the following upper bound:

$$C(L \mid n) \;\leq\; C(B) + C(L \setminus B \mid B) + \log \binom{\ell}{b} + \log(n \log n) + O(1)\ .$$

More precisely, the coding of L (given n) is composed of the binary string $S = S_1 S_2 S_3 S_4$ where:

1. S_2 is a coding for B on $C(B)$ bits.
2. S_1 the binary representation of $|S_2| = C(B)$ on exactly $\lceil \log(n \log n) \rceil$ bits. Clearly, there are at most n^ℓ sub-sequences B, thus $|S_2| \leq \ell \log n + O(1)$. And, so $|S_2|$ can be represented with exactly $\lceil \log(n \log n) \rceil$ bits exactly for n large enough. So reading the first $\lceil \log(n \log n) \rceil$ bits of S, one can then extract S_2.
3. S_3 is a coding of the ℓ-bit binary string on $\lceil \log \binom{\ell}{b} \rceil$ bits exactly. Once B is constructed, $b = |B|$ is known, so S_3 can be extracted from S after S_1 and S_2 are.
4. S_4 is a coding for $L \setminus B$ (given B).

The sub-sequence $L \setminus B$ is of length $\ell - b$, and is composed of distinct integers taken from $\{1, \ldots, n\} \setminus B$. So, once B has been constructed, it remains at most $(\ell - b)!\binom{n-b}{\ell-b}$ possible sub-sequences for $L \setminus B$. In other words, $C(L \setminus B \mid B) \leq \log\left((\ell - b)!\binom{n-b}{\ell-b}\right) + O(1)$. Therefore, we have the upper bound for the Kolmogorov Complexity of L:

$$C(L \mid n) \;\leq\; C(B) + \log\left((\ell - b)!\binom{n-b}{\ell-b}\right) + \log \binom{\ell}{b} + \log(n \log n) + O(1)\ .$$

Using the previous lower bound on $C(L \mid n)$, it follows:

$$\log\left(\ell!\binom{n}{\ell}\right) \;\leq\; \log\left((\ell - b)!\binom{n-b}{\ell-b}\binom{\ell}{b}\right) + C(B) + \log(n \log n) + O(1)\ . \tag{3}$$

Setting $t = C(B) + \log(n \log n) + O(1)$, and using the fact that $(\ell - b)!\binom{\ell}{b} = \ell!/b!$, Ineq. (3) rewrites in

$$b!\binom{n}{\ell} \;\leq\; 2^t \cdot \binom{n-b}{\ell-b}\ .$$

We have seen in the first claim that, by Ineq. (1),

$$\binom{n}{\ell} \;\le\; \binom{n-a}{\ell-a}\cdot 2^m \quad\Rightarrow\quad a \;\le\; m+\log\sqrt{2\pi n}$$

Therefore, plugging $a = b$ and $m = t - \log(b!)$, and remarking that $b! \ge (b/e)^b$, we obtain, for n large enough

$$\begin{aligned} b+\log(b!) \;\le\; t+\log\sqrt{2\pi n} \;&\Rightarrow\; b(1+\log(b/e)) \le t+\log\sqrt{2\pi n}\\ &\Rightarrow\; b\log(2b/e) \;\le\; C(B)+2\log n \end{aligned}$$

Let $C = C(B) + 2\log n$. Observe that $x/\log x \le y/\log y$ for all $2 \le x \le y$. So

$$\frac{b\log(2b/e)}{\log(b\log(2b/e))} \;\le\; \frac{C}{\log C}\;. \tag{4}$$

Note that $\log x \le 4x/e$ for every $x \ge 1$. So $b\log(2b/e) \le b\cdot 4b/e^2 = (2b/e)^2$. Hence, Ineq. (4) implies that

$$\frac{b}{2} = \frac{b\log(2b/e)}{\log((2b/e)^2)} \;\le\; \frac{b\log(2b/e)}{\log(b\log(2b/e))} \;\le\; \frac{C}{\log C} \quad\Rightarrow\quad b \;\le\; \frac{2C}{\log C}\;.$$

proving the second claim, and completing the proof of Lemma 15. □

4.3 The star network

We consider a weighted "star" with n nodes, that is a tree with root r and $n-1$ leaves labeled from 1 to $n-1$. The port numbers of the incident edges of r are fixed according to a sequence L satisfying Lemma 15 (but with parameter $n-1$ instead of n). For short, we set $\ell = |L| = \lfloor (n-1)/2 \rfloor$. More precisely, for $y \in \{1,\ldots,\ell\}$, $\mathrm{port}(r,y) = L[y]$, and all the other ports are fixed arbitrarily (but still keeping the conventions, in particular, all the ports of r are distinct and $\mathrm{port}(y,r) = 1$ for each leaf y). We put weight 1 on all edges (r,y) with $y \in \{1,\ldots,\ell\}$, and k on the other edges, where $k \ge 1$ is an integral parameter.

We assume now that R is any routing function for the star provided by any routing scheme which is name-independent in the *fixed port model*. Namely, the routing scheme is not allowed to change names of nodes and of ports. Note that we do not claim any assumption on header size. Let us denote by $M = M(n,k)$ the maximum memory requirement for R on the star. In other words, M is the smallest integer such that, for every node u, the routing function R_u can be computed by a program of size at most M bits.

4.4 The proof

The main idea is to construct all the headers and port numbers generated by the routing from r to all the destinations at distance 1, i.e., for destinations $y \in \{1,\ldots,\ell\}$. We upper bound the "total amount of information" carried by all the sequences of headers and ports up to the $(2k-1)$th first steps, and we argue that if $M < \Theta((n \log n)^{1/k})$ then the stretch factor of R is at least $2k+1$, or equivalently, if the stretch is $< 2k+1$ then $M = \Omega((n \log n)^{1/k})$. The main difficulty is to evaluate and to make precise sense of the notion "total amount of information".

For every destination y, we denote by $x_0(y), x_1(y), \ldots, x_t(y)$ the walk in the star from the source $r = x_0(y)$ to $y = x_t(y)$ induced by the routing function R. We also denote by $h_0(y), \ldots, h_t(y)$, by $q_0(y), \ldots, q_t(y)$ and by $p_1(y), \ldots, p_{t+1}(y)$ respectively the sequences of headers, input port and output port numbers generated by R from the source r to the destination y. For instance, for destination $y = 1, 2, 3, \ldots,$ we may obtain the following array of nodes:

$$\begin{array}{llllll}
x_0(1) & x_1(1) & x_2(1) & x_3(1) & & \\
x_0(2) & x_1(2) & & & & \\
x_0(3) & x_1(3) & x_2(3) & x_3(3) & x_4(3) & x_5(3) \\
\vdots & & & & &
\end{array}$$

In this example, the route from r to $y = 1$ is done in three steps (that is the walk $r, x_1(1), x_2(1) = r, 1$), the route to $y = 2$ is done directly to node 2, the route to $y = 3$ is done along 5 edges, and so on. Note that each row must begin with $x_0(y) = r$ and must end with some $x_t(y) = y$. The length of the route from r to y is then t.

Similarly, for headers and port numbers, we may obtain the arrays:

$$\begin{array}{llllll}
h_0(1) & h_1(1) & h_2(1) & h_3(1) & & \\
h_0(2) & h_1(2) & & & & \\
h_0(3) & h_1(3) & h_2(3) & h_3(3) & h_4(3) & h_5(3) \\
\vdots & & & & &
\end{array}$$

$$\begin{array}{llllll}
q_0(1) & q_1(1) & q_2(1) & q_3(1) & & \\
q_0(2) & q_1(2) & & & & \\
q_0(3) & q_1(3) & q_2(3) & q_3(3) & q_4(3) & q_5(3) \\
\vdots & & & & &
\end{array}$$

$$\begin{array}{llllll}
p_1(1) & p_2(1) & p_3(1) & 0 & & \\
p_1(2) & 0 & & & & \\
p_1(3) & p_2(3) & p_3(3) & p_4(3) & p_5(3) & 0 \\
\vdots & & & & &
\end{array}$$

We now concentrate our attention only on destinations $y \in \{1, \ldots, \ell\}$, thus all the nodes that are at distance 1 exactly from the source.

For every $i \geq 0$, let $W_i \subseteq \{1, \ldots, \ell\}$ be the subset of destinations reached by R from r after a walk of length at most $2i+1$ (all routes have odd lengths by the topology). Formally, $W_i = \{y \in \{1, \ldots, \ell\} \mid \exists j \leq i,\ x_{2j+1}(y) = y\}$. Let $i_{\max}$ be the index such that $|W_{i_{\max}}| = \ell$. By definition, if $|W_i| < \ell$, then at least one destination of $\{1, \ldots, \ell\}$ cannot be reached after $2i+1$ steps from the source. As the graph is bipartite, at least two more steps are needed resulting in a stretch $\geq 2i+3$ for R. In other words,

Claim 2. *If the stretch of R is $< 2k+1$, then $i_{\max} < k$.*

For every $i \geq 0$, let H_i be the sequence of headers obtained by extracting the ith column from the above array of headers. Note that the H_i's may have different length since the walks from r use possibly different number of edges. On the previous example,

$$\begin{array}{rllll} H_0 = & h_0(1) & h_0(2) & h_0(3) & \ldots \\ H_1 = & h_1(1) & h_1(2) & h_1(3) & \ldots \\ H_2 = & h_2(1) & h_2(3) & \ldots & \\ \vdots & & & & \end{array}$$

We define similarly the sequences Q_i and P_i. Observe that by construction, each item of P_{2i+1}, $i \geq 0$, is a port number leading from the root to a leaf of the star, whereas P_{2i+2} is composed of 0's or of 1's only. Recall that for a sequence such as P_{2i+1}, the notation P^*_{2i+1} denotes the set of elements that appear in the sequence.

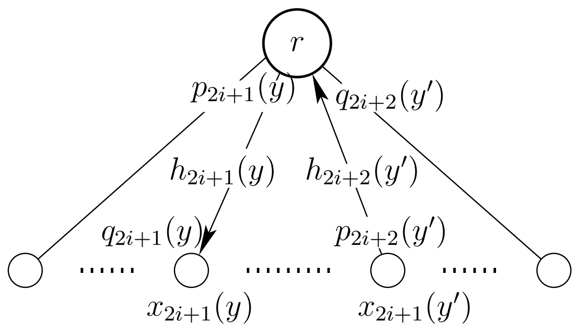

Figure 4: A step of the walks to y and y' in the star network.

Lemma 16. *For every $i \geq 0$, $|W_i| \leq \sum_{j=0}^{j \leq i} |P^*_{2j+1}|$.*

Proof. At the source, there are $|P^*_{2j+1}|$ distinct port numbers generated by R. So only at most $|P^*_{2j+1}|$ destinations can be reached at each j. So, a total of at most $\sum_{j=0}^{j \leq i} |P^*_{2j+1}|$ destinations can be reached by a walk of length at most $2i+1$. By definition of W_i, this sum must be an upper bound on $|W_i|$. □

Lemma 17. *If the stretch of R is $< 2k+1$, then $P^*_{2i+1} \subseteq L^*$ for every $i \geq 0$. Moreover, $P_{2i_{\max}+1}$ is a sub-sequence of L.*

Proof. Assume that there is some $p \in P^*_{2i+1}$ such that $p \notin L^*$. Then, the port p necessarily leads to a node $w \notin \{1,\ldots,\ell\}$. Since P_{2i+1} concerns routing from r to the nodes in $\{1,\ldots,\ell\}$, it follows that there is a walk from r to a node $y_0 \in \{1,\ldots,\ell\}$ going through

$w \notin \{1,\ldots,\ell\}$. Since the cost of the edge (r,w) is k, the resulting cost of the route from r to y_0 is at least $2k+1$: a contradiction with a stretch $< 2k+1$, r and y_0 are at distance 1.

Let Y_i be the sequence constructed by R as follows: 1) $Y_0 = 1,2,\ldots,\ell$; and 2) Y_i is the sequence obtained from Y_{i-1} by removing all the items of W_i. We have (by induction): 1) Y_i a sequence of destinations non-reached after $2i$ steps; and 2) all the items of Y_i appear in increasing order. Consider now $P_{2i_{\max}+1}$. Each port of this sequence leads to some destinations of $Y_{i_{\max}}$. By definition of $i_{\max}$, this represents the last step of the walks, and so the jth port of $P_{2i_{\max}+1}$ leads precisely to the leaf named $Y_{i_{\max}}[j]$. Because items of Y_i are in increasing order, $P_{2i_{\max}+1}$ is a sub-sequence of L. □

Lemma 18. *There is a constant c such that for every $i \geq 0$,*

1. $C(H_0, Q_0 \mid \ell) \leq c$.
2. $C(Q_{2i+2} \mid P_{2i+1}, P_{2i+2}) \leq c$
3. $C(H_{2i+1}, P_{2i+1} \mid H_{2i}, Q_{2i}) \leq M + c$.
4. $C(H_{2i+2}, P_{2i+2} \mid H_{2i+1}, P_{2i+1}, M) \leq |P^*_{2i+1}| \cdot (M+1) + c$.

Proof. By convention, $h_0(y) = y$ and $q_0(y) = 0$ for every $y \in \{1,\ldots,\ell\}$. So, $H_0 = 1,2,\ldots,\ell$, and $Q_0 = 0,\ldots,0$. Given ℓ, a trivial program of constant size suffices to construct (H_0, Q_0), proving Property 1. To prove Property 2, we remark that the sequence Q_{2i+2} can be computed by removing from P_{2i+1} the 0's entries of P_{2i+2}. For concreteness, let us consider the following example where all destinations, except $y_1,\ldots,y_6$, have been reached before $2i+1$ steps, and that P_{2i+1} and P_{2i+2} are given by:

$$\begin{array}{rcrrrrrr} & & y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ P_{2i+1} & = & 2 & 3 & 3 & 1 & 2 & 2 \\ P_{2i+2} & = & 1 & 0 & 1 & 1 & 1 & 0 \end{array}$$

It means that the walk to y_2 and y_6 have ended at step $2i+1$. Because the input and output port numbers are the same if they correspond to the same incident edge, we obtain in this example:

$$\begin{array}{rcrrrr} & & y_1 & y_3 & y_4 & y_5 \\ Q_{2i+1} & = & 2 & 3 & 1 & 2 \end{array}$$

In other words, $C(Q_{2i+2} \mid P_{2i+1}, P_{2i+2}) \leq O(1)$, proving Property 2.

The sequences H_{2i+1} and P_{2i+1} are respectively headers and output port numbers of messages that have entered and have left the source r (cf. Figure 4). Thus $(h_{2i+1}, p_{2i+1}) = R(r, h_{2i}, q_{2i})$ for all $h_{2i} \in H_{2i}$ and $q_{2i} \in Q_{2i}$. Because R_r (the function R at r) is a program of length at most M, $C(H_{2i+1}, P_{2i+1} \mid H_{2i}, Q_{2i}) \leq M + O(1)$, proving Property 3. Let us now prove Property 4. Recall that $\mathrm{port}(r,y) = L[y]$ for all $y \in \{1,\ldots,\ell\}$. Let us denote by y_p the unique destination of $\{1,\ldots,\ell\}$ such that $\mathrm{port}(r, y_p) = p$.

The sequences H_{2i+2} and P_{2i+2} are respectively headers and output port numbers of messages that have entered and have left some leaves (cf. Figure 4). The leaves concerned are the nodes y_p's such that $p \in P_{2i+1}$. The jth item of H_{2i+2} and P_{2i+2}, and thus the whole sequences, can be computed as follows: for every $j = 1, \ldots, |H_{2i+1}|$

$$(H_{2i+2}[j], P_{2i+2}[j]) \;=\; R(y_{P_{2i+1}[j]}, H_{2i+1}[j], 1) \tag{5}$$

noting that the sequences $H_{2i+1}, H_{2i+2}, P_{2i+1}, P_{2i+2}$ have same length, and that the input port of a message entering in a leaf is necessary 1.

The number of different leaves used in the calculation of Eq. (5) is $t = |P^*_{2i+1}|$. So, given P_{2i+1}, H_{2i+1}, Eq. (5) can be computed thanks to an extra table $T = (R_{y_{p(1)}}, \ldots, R_{y_{p(t)}})$ of local routing functions where $p(r)$ denotes the rth smallest port number of P_{2i+1}. The index $p(r)$, for every $r \in \{1, \ldots, t\}$, can be easily computed from P_{2i+1}.

Each entry of T is a program of length at most M. We code each of them with a binary string of $M + 1$ exactly, by padding some 0's and a 1 to the code. So the code for the rth entry, say S_r, is $0^{M-|S_r|}1S_r$ (S_r can be extracted by first reading and counting all the 0's (if any) and then by removing the first 1). Therefore, given M, T is coded with $t \cdot (M + 1) = |P^*_{2i+1}| \cdot (M + 1)$ bits. Overall, we have shown that $C(H_{2i+2}, P_{2i+2} \mid H_{2i+1}, P_{2i+1}, M) \leq |P^*_{2i+1}| \cdot (M + 1) + O(1)$, proving Property 4 and completing the proof of Lemma 18. □

We have now all the material to prove Theorem 2. Intuitively, during the walks to the destinations, the Kolmogorov Complexity of the header and port sequences increases with the number of steps. However, this increment is controlled by Lemma 18, and is mainly limited by Eq. 4. If the stretch is $< 2k + 1$, then the port sequence is a subset of L^*, and thus Lemma 15 indicates that, in this case, the number of distinct ports in that sequence ($|P^*_{2i+1}|$) and its Kolmogorov Complexity are closely related. All together this leads to the fact that the number of destinations reachable in at most $2i + 1$ steps is $|W_i| = O(M^{i+1})$. In particular, for the very last step $i = i_{\max}$, $|W_{i_{\max}}| = n/2 = O(M^{i_{\max}+1})$ implying that $M = \Omega(n^{i_{\max}+1}) = \Omega(n^{1/k})$ since by Claim 2, $i_{\max} < k$. The claimed lower bound of $\Omega((n \log n)^{1/k})$ is a refinement of the proof process sketched above, and it is obtained by analyzing in a different way the very last step $i_{\max}$.

We can now prove Theorem 2 that is:

Theorem 2. *For each integer $k \geq 1$ there is an n-node star with edge weights 1 or k for which every single-source name-independent routing scheme with less than $\frac{1}{9}(n \log n)^{1/k}$ bits of memory requirements has stretch factor at least $2k + 1$.*

Proof. Assume, by way of contradiction, that the stretch factor of R is $< 2k + 1$. We first need to prove some technical results which is standard for reader familiar with the Kolmogorov Complexity. Some objects used in the proof are determined by two or more other objects, and we will need to code them in a *self-delimiting* way, that is using prefix-free codes. Indeed, this will allow us to concatenate and to compose them. The prefix-free codes guarantees that each part can be then extracted from the whole concatenation.

Claim 3. *Any non-null integer w has a self-delimiting coding of length less than $2\log w$ bits. In particular, any object X with $C(X) \geq 1$ has a self-delimiting coding of length at most $C(X) + 2\log C(X)$.*

Proof. Let B be the standard binary string representation of w. As $w \geq 1$, B begins with a 1 followed by $\lceil \log w \rceil - 1 = b$ bits. If $b = 0$, i.e., $w = 1$ then w is coded by a single 0. If $b \geq 1$, then the code for w is the word $0^{b-1}B$. We check that w can be self-extracted by first reading all the 0's of the code (it will stop at the first 1 of B). Its length is $(b-1) + 1 + b = 2(\lceil \log w \rceil - 1) < 2\log w$. The self-delimiting coding of any object X consists in preceding any coding of X of length $C(X)$ with a self-delimiting coding of the integer $w = C(X)$. $\square$

Claim 4. *There is a constant c such that, for all objects X and Y, $C(X,Y) \leq C(X \mid Y) + C(Y) + 2\min\{\log C(X \mid Y), \log C(Y)\} + c$.*

Proof. Let P be a minimal length program that, given Y as input, outputs X. The length of P is $C(X \mid Y)$. The couple (X,Y) can be represented by the concatenation of a self-delimiting coding for Y followed by P, or alternatively, by a self-delimiting coding for P followed by Y. This choice is coded by an extra bit. By Claim 3, the length of this coding is $C(X \mid Y) + C(Y) + 2\min\{\log C(X \mid Y), \log C(Y)\} + 1$, proving the claim. $\square$

Observe that, since $C(X \mid Y) \leq C(X)$, Claim 4 directly implies that $C(X,Y) \leq C(X) + C(Y) + 2\min\{\log C(X), \log C(Y)\} + c$.

Claim 5. *There is a constant c such that, for all objects X, Y, Z, $C(X \mid Z) \leq C(X \mid Y) + C(Y \mid Z) + 2\min\{\log C(X \mid Y), \log C(Y \mid Z)\} + c$.*

Proof. We need to construct a shortest program P that outputs X with input Z. For such a program one can concatenate and combine two minimal programs: one that outputs Y from Z, say program P_1 of length $C(Y \mid Z)$, and one that outputs X from Y, say program P_2 of length $C(X \mid Y)$. Clearly, the length of P is $C(X \mid Z) \leq C(P_1, P_2) + O(1)$, which is bounded by $C(Y \mid Z) + C(X \mid Y) + 2\min\{\log C(Y \mid Z), \log C(X \mid Y)\} + O(1)$ by Claim 4, completing the proof of the claim. $\square$

Plugging $Z = \varepsilon$ (the empty string) in Claim 5, we have that $C(X) \leq C(X \mid Y) + C(Y) + 2\min\{\log C(X \mid Y), \log C(Y)\} + c$. For notational convenience, we assume hereafter along the proof that n, ℓ, k and M are given parameters. For instance, when $C(X \mid Y)$ is used in a formula, it stands for $C(X \mid Y, n, \ell, k, M)$.

Combining Claim 4 with Property 4 of Lemma 18, we obtain for every $i \geq 0$:

$$\begin{aligned}
C(H_{2i+2}, P_{2i+2}) \;\leq\;& C(H_{2i+2}, P_{2i+2} \mid H_{2i+1}, P_{2i+1}) + C(H_{2i+1}, P_{2i+1}) && (6)\\
& +2\log C(H_{2i+2}, P_{2i+2} \mid H_{2i+1}, P_{2i+1}) + O(1) && (7)\\
\leq\;& |P^*_{2i+1}| \cdot (M+1) + C(H_{2i+1}, P_{2i+1}) && (8)\\
& +2\log\left(|P^*_{2i+1}| \cdot (M+1)\right) + O(1) && (9)
\end{aligned}$$

Since the stretch is $< 2k+1$, one can apply Lemma 15 to P^*_{2i+1}, and thus we have $|P^*_{2i+1}| \leq C(P^*_{2i+1}) + 2\log n$. Clearly, P^*_{2i+1} can be computed given P_{2i+1}, i.e., $C(P^*_{2i+1}) \leq C(P_{2i+1}) + O(1)$. Hence,

$$|P^*_{2i+1}| \ \leq \ C(P^*_{2i+1}) + 2\log n + O(1) \ .$$

We have also that $|P^*_{2i+1}| \leq n/2$ so that $|P^*_{2i+1}| \cdot (M+1) \leq nM$. Combining these with Ineq. (9), we obtain:

$$\begin{aligned} C(H_{2i+2}, P_{2i+2}) \ &\leq \ (C(P_{2i+1}) + 2\log n + O(1)) \cdot (M+1) \\ &\quad + C(H_{2i+1}, P_{2i+1}) + 2\log(nM) + O(1) \\ &\leq \ C(H_{2i+1}, P_{2i+1}) + (M+1) \cdot (C(P_{2i+1}) + 2\log n) + O(M + \log n) \end{aligned}$$

In other words, for every $i \geq 1$, we have:

$$C(H_{2i}, P_{2i}) \ \leq \ C(H_{2i-1}, P_{2i-1}) + (M+1) \cdot (C(P_{2i-1}) + 2\log n) + O(M + \log n) \quad (10)$$

Combining Properties 2 and 3 of Lemma 18 with Claim 5, we have that for every $i \geq 1$,

$$C(H_{2i+1}, P_{2i+1} \mid H_{2i}, P_{2i}, P_{2i-1}) \ \leq \ M + O(1)$$

Therefore, applying several times Claim 4,

$$\begin{aligned} C(H_{2i+1}, P_{2i+1}) \ &\leq \ C(H_{2i+1}, P_{2i+1} \mid H_{2i}, P_{2i}, P_{2i-1}) + C(H_{2i}, P_{2i}, P_{2i-1}) \\ &\quad + 2\log C(H_{2i+1}, P_{2i+1} \mid H_{2i}, P_{2i}, P_{2i-1}) + O(1) \\ &\leq \ C(H_{2i}, P_{2i}, P_{2i-1}) + M + 2\log M + O(1) \\ &\leq \ C(H_{2i}, P_{2i}) + C(P_{2i-1}) + 2\log C(P_{2i-1}) + O(M) \end{aligned}$$

Observe that $C(P_{2i-1}) \leq n \log n + O(1)$, since P_{2i-1} is a sequence of at most ℓ integers taken from $\{1, \ldots, n-1\}$. So, for every $i \geq 1$,

$$\begin{aligned} C(H_{2i+1}, P_{2i+1}) \ &\leq \ C(H_{2i}, P_{2i}) + C(P_{2i-1}) + 2\log(n\log n) + O(M) \quad &(11) \\ &\leq \ C(H_{2i}, P_{2i}) + C(P_{2i-1}) + O(M + \log n) \quad &(12) \end{aligned}$$

Plugging Ineq. (10) in Ineq. (12), we obtain for every $i \geq 1$:

$$\begin{aligned} C(H_{2i+1}, P_{2i+1}) \ &\leq \ C(H_{2i-1}, P_{2i-1}) + (M+1) \cdot (C(P_{2i-1}) + 2\log n) \\ &\quad + C(P_{2i-1}) + O(M + \log n) \\ &\leq \ C(H_{2i-1}, P_{2i-1}) + (M+2) \cdot (C(P_{2i-1}) + 2\log n) + O(M + \log n) \\ &\leq \ C(H_1, P_1) + \sum_{j=1}^{i} ((M+2) \cdot (C(P_{2j-1}) + 2\log n) + O(M + \log n)) \end{aligned}$$

Combining Claim 4 with Property 3 (for $i = 0$), and Property 1 of Lemma 18 we have:

$$\begin{aligned} C(H_1, P_1) \ &\leq \ C(H_1, P_1 \mid H_0, Q_0) + C(H_0, Q_0) + 2\log C(H_0, Q_0) + O(1) \quad &(13) \\ &\leq \ M + O(1) \quad &(14) \end{aligned}$$

Note that there is a program of length $C(A,B)$ that prints out A and B, and can tell them apart. In particular, $C(A) \leq C(A,B)+O(1)$. It follows that $C(P_{2i+1}) \leq C(H_{2i+1}, P_{2i+1})+O(1)$, thus, for a constant c large enough, we have:

$$\begin{aligned}
C(P_{2i+1}) + 2\log n &\leq (M+2)\sum_{j=1}^{i} \left(C(P_{2j-1}) + 2\log n + c(M+\log n)\right) && (15)\\
&\leq (1+\varepsilon)(M+2)\sum_{j=1}^{i} \left(C(P_{2j-1}) + 2\log n\right) && (16)
\end{aligned}$$

where ε is chosen such that $c(M+\log n) \leq \varepsilon(M+2)(2\log n)$. Note that $\varepsilon = \Omega(1/\log n + 1/M)$. Setting $C_j = C(P_{2j-1}) + 2\log n$, Ineq. (16) rewrites in,

$$\forall\, i \geq 1, \quad C_{i+1} \;\leq\; (1+\varepsilon)(M+2)\sum_{j=1}^{i} C_j \qquad (17)$$

Claim 6. *For every $i \geq 0$,*

$$C_{i+1} \;\leq\; (\alpha(M+2))^{i+1} \quad \textit{where } \alpha = 1+\varepsilon+\frac{1}{M+2}\;.$$

Proof. By induction. Because $C(P_1) \leq C(H_1, P_1) + O(1)$, Ineq. (14) yields $C(P_1) \leq M+O(1)$. Hence $C(P_1)+2\log n \leq (1+\varepsilon)(M+2)$ by the choice of ε. So, $C_1 \leq (1+\varepsilon)(M+2)$ and Claim 6 holds for $i=0$. Assume it holds for every $1 \leq j \leq i$. We have:

$$\begin{aligned}
(1+\varepsilon)(M+2)\sum_{j=1}^{i} C_j &\leq (1+\varepsilon)(M+2)\cdot C_1 \cdot \frac{(\alpha(M+2))^i - 1}{\alpha(M+2)-1}\\
&\leq ((1+\varepsilon)(M+2))^2 \cdot \frac{(\alpha(M+2))^i - 1}{(1+\varepsilon)(M+2)}\\
&< (1+\varepsilon)\alpha^i \cdot (M+2)^{i+1}\\
&\leq \left(\alpha - \frac{1}{M+2}\right)\alpha^i \cdot (M+2)^{i+1} \;<\; \alpha^{i+1}(M+2)^{i+1}
\end{aligned}$$

Therefore, by Ineq. (17), $C_{i+1} \leq (1+\varepsilon)(M+2)\sum_{j=1}^{i} C_j \leq (\alpha(M+2))^{i+1}$ completing the proof Claim 6. □

The stretch is $< 2k+1$, thus by Lemma 15 and Claim 6,

$$\begin{aligned}
\sum_{j=0}^{i} |P^*_{2j+1}| &\leq \sum_{j=0}^{i} (C(P^*_{2j+1}) + 2\log n)\\
&\leq \sum_{j=0}^{i} (C(P_{2j+1}) + 2\log n + O(1))\\
&= O(i) + \sum_{j=1}^{i+1} C_j
\end{aligned}$$

From the proof of Claim 6, we have directly that

$$\sum_{j=1}^{i+1} C_j \;<\; \frac{(\alpha(M+2))^{i+2}}{(1+\varepsilon)(M+2)} \;=\; \frac{\alpha^{i+2}}{1+\varepsilon}\cdot(M+2)^{i+1}$$

By Lemma 16, for M large enough, i.e., $M=\Omega(\log n)$, $\alpha>1$, and $\varepsilon>0$.

$$\forall\, i\geq 0,\quad |W_i| \;\leq\; \sum_{j=0}^{i}|P^*_{2j+1}| \;<\; O(i)+\frac{\alpha^{i+2}}{1+\varepsilon}\cdot(M+2)^{i+1} \;<\; \alpha^{i+2}\cdot(M+2)^{i+1}$$

In particular, for $i=i_{\max}$,

$$\ell=|W_i| \;\leq\; \sum_{j=0}^{i}|P^*_{2j+1}| = |P^*_{2i_{\max}+1}| + \sum_{j=0}^{i_{\max}-1}|P^*_{2j+1}| \;<\; |P^*_{2i_{\max}+1}| + \alpha^{i+2}\cdot(M+2)^{i+1}$$

from Lemma 17, $P_{2i_{\max}}$ is a sub-sequence of L, and since L is composed of distinct elements, $|P^*_{2i_{\max}+1}|=|P_{2i_{\max}+1}|$, and so Lemma 15 can be applied. It turns out:

$$\ell \;<\; \frac{2C(P_{2i_{\max}+1})+4\log n}{\log(C(P_{2i_{\max}+1})+2\log n)} + \alpha^{i_{\max}+1}\cdot(M+2)^{i_{\max}}$$

Noting that $2C(P_{2i_{\max}+1})+4\log n \;=\; 2C_{i_{\max}}$, one can applied Claim 6, and hence (as $\alpha^{i_{\max}+1}\geq 1$):

$$\ell \;<\; \frac{2\alpha^{i_{\max}+1}\cdot(M+2)^{i_{\max}+1}}{\log((M+2)^{i_{\max}+1})} + \alpha^{i_{\max}+1}\cdot(M+2)^{i_{\max}}$$

Now, because the stretch factor of R is $<2k+1$, $i_{\max}\leq k-1$ by Claim 2. Thus,

$$\ell \;<\; \frac{2\alpha^{k}\cdot(M+2)^{k}}{\log((M+2)^{k})} + \alpha^{k}\cdot(M+2)^{k-1}$$

because for $M=\Omega(\log n)$ large enough, one can choose $\varepsilon=c/\log n$, for a suitale constant c. Hence $\alpha^k\leq(1+O(1/\log n))^k\leq 2$ for every $k\geq 1$. Because, $\ell\sim n/2$, it follows that

$$M \;\geq\; \left(\frac{\ell}{4}\log(\ell/4)\right)^{1/k} - 2 \;\geq\; \frac{1}{9}\,(n\log n)^{1/k}$$

for n large enough, that completes the proof of Theorem 2. □

5 Conclusion

Our routing schemes can be adopted to work on strongly connected directed graphs, this extension will appear in the full paper. Although our upper and lower bounds are asymptotically optimal there remains the challenge of finding the exact constants. Specifically, what is the exact lowest stretch attainable with a $\widetilde{O}(n^{1/k})$ memory? Even for the labeled case, the bound is not tight.

References

[1] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Routing with improved communication-space trade-off. In 18^{th} *International Symposium on Distributed Computing (DISC)*, volume 3274 of Lecture Notes in Computer Science, pages 305–319. Springer, October 2004.

[2] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. In 16^{th} *Annual ACM Symposium on Parallel Algorithms and Architecture (SPAA)*, pages 20–24. ACM Press, July 2004.

[3] Noga Alon, Shlomo Hoory, and Nathan Linial. The Moore bound for irregular graphs. *Graph and Combinatorics*, 18(1):53–57, 2002.

[4] Marta Arias, Lenore J. Cowen, Kofi Ambrose Laing, Rajmohan Rajaraman, and Orjeta Taka. Compact routing with name independence. In 15^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 184–192. ACM Press, June 2003.

[5] Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Compact distributed data structures for adaptive routing. In 21^{st} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 479–489. ACM Press, May 1989.

[6] Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Improved routing strategies with succinct tables. *Journal of Algorithms*, 11(3):307–341, 1990.

[7] Baruch Awerbuch and David Peleg. Sparse partitions. In 31^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 503–513. IEEE Computer Society Press, October 1990.

[8] Baruch Awerbuch and David Peleg. Routing with polynomial communication-space trade-off. *SIAM Journal on Discrete Mathematics*, 5(2):151–162, May 1992.

[9] Harry Buhrman, Jaap-Henk Hoepman, and Paul Vitányi. Space-efficient routing tables for almost all networks and the incompressibility method. *SIAM Journal on Computing*, 28(4):1414–1432, 1999.

[10] J. Lawrence Carter and Mark N. Wegman. Universal hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[11] Lenore J. Cowen. Compact routing with minimum stretch. *Journal of Algorithms*, 38:170–183, 2001.

[12] Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch factor. *Journal of Algorithms*, 46:97–114, 2003.

[13] Paul Erdös. Extremal problems in graph theory. In *Publ. House Cszechoslovak Acad. Sci., Prague*, pages 29–36, 1964.

[14] Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of Lecture Notes in Computer Science, pages 757–772. Springer, July 2001.

[15] Pierre Fraigniaud and Cyril Gavoille. A space lower bound for routing in trees. In 19^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2285 of Lecture Notes in Computer Science, pages 65–75. Springer, March 2002.

[16] Cyril Gavoille. Routing in distributed networks: Overview and open problems. *ACM SIGACT News - Distributed Computing Column*, 32(1):36–52, March 2001.

[17] Cyril Gavoille and Marc Gengler. Space-efficiency of routing schemes of stretch factor three. *Journal of Parallel and Distributed Computing*, 61:679–687, 2001.

[18] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Journal of Distributed Computing*, 16:111–120, May 2003. PODC 20-Year Special Issue.

[19] Cyril Gavoille and Stéphane Pérennès. Memory requirement for routing in distributed networks. In 15^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 125–133. ACM Press, May 1996.

[20] Evangelos Kranakis and Danny Krizanc. Lower bounds for compact routing. In 13^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1046 of Lecture Notes in Computer Science, pages 529–540. Springer-Verlag, February 1996.

[21] Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. In 45^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 434–443. IEEE Computer Society Press, October 2004.

[22] Kofi Ambrose Laing. Brief announcement: name-independent compact routing in trees. In 23^{rd} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 382–382. ACM Press, July 2004.

[23] Felix Lazebnik, Vasiliy A. Ustimenko, and Andrew J. Woldar. A new series of dense graphs of high girth. *Bulletin of the American Mathematical Society (New Series)*, 32(1):73–79, January 1995.

[24] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications (second edition)*. Springer-Verlag, 1997.

[25] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[26] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[27] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, July 1989.

[28] Mikkel Thorup and Uri Zwick. Compact routing schemes. In 13^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10. ACM Press, July 2001.

[29] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, January 2005.