Space-Stretch Tradeoff in Routing Revisited

Anatoliy Zinovyev
Boston University, MA, USA

Abstract

We present several new proofs of lower bounds for the space-stretch tradeoff in labeled network routing.

First, we give a new proof of an important result of Cyril Gavoille and Marc Gengler that any routing scheme with stretch < 3 must use $\Omega(n)$ bits of space at some node on some network with n vertices, even if port numbers can be changed. Compared to the original proof, our proof is significantly shorter and, we believe, conceptually and technically simpler. A small extension of the proof can show that, in fact, any constant fraction of the n nodes must use $\Omega(n)$ bits of space on some graph.

Our main contribution is a new result that if port numbers are chosen adversarially, then stretch $< 2k + 1$ implies some node must use $\Omega\left(n^{\frac{1}{k}} \log n\right)$ bits of space on some graph, assuming a girth conjecture by Erdős.

We conclude by showing that all known methods of proving a space lower bound in the labeled setting, in fact, require the girth conjecture.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Data structures design and analysis; Mathematics of computing → Discrete mathematics

Keywords and phrases Compact routing, labeled network routing, lower bounds

Digital Object Identifier 10.4230/LIPIcs.DISC.2022.37

1 Introduction

Network routing is an important subject in the field of distributed algorithms. Given a network of nodes (routers) connected arbitrarily, the goal is to be able to transmit information between any two nodes in the network. A network data packet must usually traverse multiple routers in order to reach its destination, and these intermediate routers must be able to make a decision as to where to forward the packet next based only on the packet and some local information. This local information is usually referred to as a routing table or, more precisely, a routing program, and one of the goals is to bound its size. This is particularly important in practice since the memory inside a router must be fast and is therefore expensive. Another goal one might pursue is the quality of the routes chosen by the routers. The route quality is commonly characterized by the stretch factor defined as the ratio between the length of the route and the shortest distance between the two nodes. In practice, it corresponds to latency in a network that does not experience congestion.

In this paper we consider the problem of labeled routing: what is the best possible tradeoff between routing program sizes and the stretch factor given that we are allowed to assign arbitrary (but not long) labels to nodes which the sender must include in the routing header? Although this model is of little practical interest since one doesn't want to reconfigure the labels each time the network changes, labeled routing schemes often serve as an ingredient in name-independent routing: first a label is retrieved from a distributed dictionary after which a labeled routing scheme is used [12, 5, 4, 1]. Additionally, the lower bounds for the labeled routing setting imply the same lower bounds for the name-independent setting, where node labels are chosen adversarially.

1.1 Some general notation

For a natural number n, define $[n] = \{1, 2, ..., n\}$. For any vertex v, denote $\mathcal{N}(v)$ to be the set of neighbors of v. Define $\deg(v) = \left|\mathcal{N}(v)\right|$, the degree of v.

1.2 Model

We deal with undirected connected graphs G with n nodes (routers). Let $G = (V, E)$ where $V = [n]$. Each $x \in V$ possesses a unique label (name) $l_x \in \{0, 1\}^*$, as well as a routing program R_x. The goal of a (distributed) routing algorithm is to route data from any node x to any node y given only the destination's label l_y. Routers are allowed to attach an arbitrary header to messages and modify the header at each hop. Each router x has network ports numbered $\{1, 2, ..., \deg(x)\}$ to which edges are attached. This port-edge relation is a bijection: each edge is assigned exactly one port and each port is assigned exactly one edge. For $x \in V$ and $1 \leq t \leq \deg(x)$ define $\tau_x(t) = y$ where y is the node such that $\{x, y\}$ is the edge assigned to port t at node x. So, $\tau_x : [\deg(x)] \to \mathcal{N}(x)$ is a bijection. The routing program R_x at node x accepts the header of the incoming message and the incoming port number as input and outputs the new header and the outgoing port number: if $R_x(h, q) = (h', p)$, a message with header h that comes in through port q leaves the router with header h' through port p.

Formally, a routing scheme for G with node labels $\mathcal{L} = (l_1, l_2, ..., l_n)$ and port functions $\mathcal{T} = (\tau_1, \tau_2, ..., \tau_n)$ is a collection of programs $\mathcal{R} = (R_1, R_2, ..., R_n)$ such that for all $x, y \in V$ there exists a finite walk $x = v_0, v_1, ..., v_t = y$ in G, a sequence $h_0, h_1, ..., h_t$ of headers, and two sequences $q_0, ..., q_t$ and $p_1, ..., p_{t+1}$ of port numbers such that:

- for all $0 \leq i < t$:
  - $R_{v_i}(h_i, q_i) = (h_{i+1}, p_{i+1})$
  - $\tau_{v_i}(p_{i+1}) = v_{i+1}$
  - $\tau_{v_{i+1}}(q_{i+1}) = v_i$
- $h_0 = l_y$
- $q_0 = 0$
- $R_{v_t}(h_t, q_t) = (\epsilon, 0)$

This model can take different variations depending on whether the node labels and the port assignment are adversarially chosen or not. In the most adversarial setting, both assignments are given as part of the input and the goal is to construct a correct routing scheme. In the least adversarial setting, only the graph G is given, and the goal is to generate a routing scheme along with node labels and a port assignment. Models where only one assignment is adversarial can also be considered. Generally, minimizing the size of routing programs, node labels and headers is desirable. In this paper, we work with two models: one has non-adversarial port assignment, and the other has adversarial port assignment. Both, however, are models for labeled routing; i.e., node labels are non-adversarial.

Finally, we formally define routing stretch. We say a routing scheme has stretch s if for all x, y, $d_R(x, y) \leq s \cdot d_G(x, y)$ where $d_R(x, y)$ is the length of the path from x to y taken by the routing scheme, and $d_G(x, y)$ is the distance between x and y in G.

1.3 Known results

Many routing schemes have been proposed in the literature. The state of the art for labeled routing includes [17] by Thorup and Zwick who for every integer $k \geq 1$ describe a scheme that uses $\tilde{O}\left(n^{\frac{1}{k}}\right)$ bits of space at every node and has stretch $4k - 5$, [8] by Chechik who shows a scheme using $\tilde{O}\left(n^{\frac{1}{k}} \log D\right)$ bits of space at every node and having stretch $c \cdot k$ for

some $c < 4$ for sufficiently large integer k, and [15] by Roditty and Tov who show a routing scheme using $\tilde{O}\left(\frac{1}{\epsilon} n^{\frac{1}{k}} \log D\right)$ bits of space at each node and having stretch $4k - 7 + \epsilon$ for every integer $k \geq 1$; D is defined to be the normalized network diameter.

Surprisingly, there are also name-independent routing schemes with similar characteristics. An optimal stretch-3 name-independent routing scheme that stores $\tilde{O}\left(\sqrt{n}\right)$ bits at each node is known [4]. Additionally, [3] constructs a routing scheme with $O(k)$ stretch that uses $\tilde{O}\left(n^{\frac{1}{k}}\right)$ bits of space at each node.

Lower bounds for routing schemes also exist and are the focus of this paper. The first such lower bound appeared in the work of Peleg and Upfal [14] who showed that for any $s \geq 1$, any stretch-s routing scheme with adversarial port assignment on n nodes must use a total of $\Omega\left(n^{1+\frac{1}{s+2}}\right)$ bits, and thus $\Omega\left(n^{\frac{1}{s+2}}\right)$ bits at some node. Gavoille and Perennes [11] prove that $\Omega\left(n^2 \log n\right)$ total bits is needed for shortest path routing, even when port numbers can be chosen by the designer; however, node labels are assumed to be $[n]$. Buhrman et al. [7] explore upper and lower bounds under various routing models. In the standard model, that we consider here, they prove that shortest path routing requires $\Omega\left(n^2\right)$ bits of total state, even when ports can be chosen by the designer, and labels can be arbitrary (short) bit strings. Gavoille and Gengler [10] achieve the same lower bound under the same model, but for any stretch $s < 3$. Finally, Thorup and Zwick [17] claim a lower bound of $\Omega\left(n^{1+\frac{1}{k}}\right)$ total bits of space for stretch $s < 2k + 1$ when the port assignment is adversarial, but their proof idea does not seem to work in the standard model. We discuss their claim and proof idea in the next subsection, and present a proof which works under the standard model, with a $\log n$ factor improvement. Table 1 demonstrates known results for labeled routing. Note that a lower bound that works for the non-adversarial port assignment implies the same lower bound for the adversarial port assignment.

Table 1 Known lower bounds for labeled routing.

(a) Non-adversarial (+ adversarial) ports.

Work	*Stretch*	*Total memory (bits)*	*Local memory (bits)*	*Notes*
Gavoille and Perennes [11]	$< 5/3$	$\Omega\left(n^2 \log n\right)$	$\Omega(n \log n)$ on $\Omega(n)$ nodes	Node labels are $[n]$
Buhrman et al. [7]	1	$\Omega\left(n^2\right)$	$\Omega(n)$ on $\Omega(n)$ nodes	
Gavoille and Gengler [10]	< 3	$\Omega\left(n^2\right)$	$\Omega(n)$	
This paper	< 3	$\Omega\left(n^2\right)$	$\Omega(n)$ on cn nodes, $\forall 0 < c < 1$	

(b) Adversarial ports.

Work	*Stretch*	*Total memory (bits)*	*Local memory (bits)*	*Notes*
Peleg and Upfal [14]	$s \geq 1$	$\Omega\left(n^{1+\frac{1}{s+2}}\right)$	$\Omega\left(n^{\frac{1}{s+2}}\right)$	
Thorup and Zwick [17]	$< 2k + 1$	$\Omega\left(n^{1+\frac{1}{k}}\right)$	$\Omega\left(n^{\frac{1}{k}}\right)$	Works in a different model
This paper	$< 2k + 1$	$\Omega\left(n^{1+\frac{1}{k}} \log n\right)$	$\Omega\left(n^{\frac{1}{k}} \log n\right)$	

All known lower bound proofs for stretch $s > 1$ (Peleg and Upfal [14], Thorup and Zwick [17], our proof in section 3) rely on dense graphs of large girth. In an unweighted graph where all cycles have length $> s + 1$, routing with stretch s between neighbor vertices x and y must involve the edge $\{x, y\}$. Thus, x must "know" its neighbor y, and this is what proofs rely on. Currently, however, the existence of dense graphs of large girth is an open question. It is worth noting that Abraham et al. [2] overcome this conjecture in the name-independent model and unconditionally show a $\Omega\left((n \log n)^{\frac{1}{k}}\right)$-bit local memory requirement if weighted networks are allowed.

1.4 Our contribution

1.4.1 Lower bounds with non-adversarial port numbers

Our first contribution is a simpler proof of the result of Gavoille and Gengler ([10], with the full proof in their research report [9]) that shows that any routing scheme with stretch < 3 must use a total of $\Omega(n^2)$ bits of routing state, and thus $\Omega(n)$ bits at some node, on some network with n nodes, even if nodes and port numbers are allowed to be renamed. Our proof, presented in section 2, is three times shorter and, we believe, significantly simpler. This is due to a simpler class of graphs considered, easier reasoning and calculations.

With a bit of additional effort, we are also able to show that, in fact, any constant fraction of the n nodes must use $\Omega(n)$ bits of space on some graph. Previously, similar results were obtained in [11] and [7] for shortest path routing.

[11] proves a total memory requirement of $\Omega(n^2 \log n)$ bits with local memory requirement of $\Omega(n \log n)$ bits on $\Omega(n)$ nodes. We note that $\Omega(n^2 \log n)$-bit total memory requirement implies $\Omega(n \log n)$ bits on $\Omega(n)$ nodes using the following informal generic argument. Suppose the total memory requirement is $c_0 n^2 \log n$ bits, the shortest path routing table can be described in $c_1 n \log n$ bits at each node, and there are x nodes that store $> c_2 n \log n$ bits. Then $x \cdot c_1 n \log n + (n - x) \cdot c_2 n \log n \geq c_0 n^2 \log n$ which implies $x \geq \frac{c_0 - c_2}{c_1 - c_2} n$. Setting $c_2 = \frac{c_0}{2}$ achieves the result. The same reasoning can be used to show that the $\Omega(n^2)$ total memory proved implies $\Omega(n)$ bits at $\Omega(\frac{n}{\log n})$ nodes, but this is the best possible generic argument. Indeed, you could have $O(\frac{n}{\log n})$ nodes storing $O(n \log n)$ bits and all other nodes storing 0 bits. To show that more nodes must store $\Omega(n)$ bits, one needs a deeper argument.

[7] proves a local memory requirement of $\Omega(n)$ bits on $\Omega(n)$ nodes which can easily be extended to any constant fraction of all nodes. It uses a Kolmogorov random graph and assuming too many nodes have small routing programs reaches a contradiction.

It also seems that Gavoille and Gengler's proof [10] can be extended to show a local memory requirement of $\Omega(n)$ bits on any constant fraction of all nodes. We provide an extension in the context of our proof for completeness.

1.4.2 Lower bounds with adversarial port numbers

Our second contribution is a proof of a claim by Thorup and Zwick that if ports are assigned adversarially, for any integer $k \geq 1$, any labeled routing scheme with stretch $< 2k + 1$ requires a total of $\Omega(n^{1+\frac{1}{k}})$ bits of space, and so $\Omega(n^{\frac{1}{k}})$ bits of space at some node, in the worst case, assuming a well-known conjecture by Erdős regarding the existence of dense graphs with large girth. This claim deserves special discussion.

In [17], Thorup and Zwick make this claim, deferring the full proof to the full version of the paper which never appeared. It was stated, however, that the result should follow easily from their other paper [16] about compact distance oracles. A compact distance oracle is a small data structure for a graph that for any vertices u and v is able to return an approximate distance (up to some constant factor) between them. Their lower bound for distance oracles is now a standard incompressibility argument, perhaps first introduced by Matoušek [13], and it goes as follows. Take a graph with m edges and girth $2k + 2$ and consider all 2^m subgraphs. For any two subgraphs, there must be an edge $\{u, v\}$ in one that is absent in the other. If the distance oracle satisfies stretch $< 2k + 1$, then it must return distance $[1, 2k + 1)$ for u, v in the first graph, and distance $\geq 2k + 1$ in the second graph. Thus, each subgraph must have a distinct distance oracle data structure, one of them of size m bits. So, it appears that the lower bound proof for routing that Thorup and Zwick had in mind is assuming the existence of a routing scheme with stretch $< 2k + 1$ of total size $< m$ bits, and constructing

distance oracle data structures: given a graph with a compact routing scheme, construct a program that for any vertices u, v traces the route from u to v returned by the routing scheme and reports the length of the route. Clearly, if a routing scheme can be encoded with $< m$ bits, then $< 2^m$ routing schemes are needed to satisfy all graphs, and $< 2^m$ distance oracle programs are needed, a contradiction. The problem with this argument is that one cannot trace the route from u to v given the routing scheme alone. Each routing program returns the port toward the next node, but we do not know what the next node is, and encoding the port assignment takes at least m bits which would break the proof.

Obviously, however, this argument works in the model where the port must equal to the node label it leads to, or if the ports are $\log n$ bit strings chosen adversarially, but this is less natural than providing the router with a port in $\{1, ..., |\mathcal{N}(v)|\}$, and is perhaps more suited for wireless networks. Given that this lower bound was mentioned in the context of the standard model in so many works ([12, 2, 5, 4, 1, 8]), we set out to close this gap and prove the result in the standard, more natural, model. Instead of considering many graphs of large girth, we fix one such dense graph, consider all possible port assignments, and show that a single routing scheme cannot satisfy many of them; thus many routing schemes are needed. Our bound is also slightly stronger: assuming Erdős' conjecture, stretch $< 2k + 1$ requires a total of $\Omega\left(n^{1+\frac{1}{k}} \log n\right)$ bits of space. This implies that the stretch-3 routing scheme of Thorup and Zwick [17] has optimal up to $\sqrt{\log n}$ factor per-node space requirement.

To the best of our knowledge, the best previous lower bound for stretch $s \geq 3$ routing with adversarial ports in the standard model is given by Peleg and Upfal who showed a total memory requirement of $\Omega\left(n^{1+\frac{1}{s+2}}\right)$ [14]. Their proof is based on a probabilistic algorithm for constructing dense graphs with large girth. In contrast, our proof decouples the existence of such graphs and the lower bound argument, which lets us use the best known results regarding the existence of such graphs. In particular, the graph construction in [14] can be used in conjunction with the proof in this paper to obtain the lower bound result in [14], but better graph constructions are available.

1.4.3 Known techiques require girth conjecture

Finally, in section 4 we show that all known methods for proving a space lower bound, that is, finding the number of distinct routing schemes necessary, actually require the existence of dense graphs with large girth.

2 Lower bounds with non-adversarial port numbers

We are interested in the complexity of a routing scheme defined by $\max\{|R_v| : v \in V\}$ – the size of the largest routing algorithm.

The idea of the lower bound, as in [10], is to consider a family of graphs of girth 4 (where all cycles have length ≥ 4) and show that $2^{\Omega(n^2)}$ configurations of routing programs $\mathcal{R}$ are needed to satisfy all graphs in the family with stretch < 3. From this it follows that at least one routing program must have size $\Omega(n)$ bits, as shown by the following lemma.

▶ **Lemma 1.** *Let n be a positive integer, and let $S \subseteq \left(\{0,1\}^*\right)^n$ be a collection of n-tuples of binary strings with $|S| > 0$. Then some string must have length $\geq \frac{\log|S|}{n} - 1$.*

Proof. Suppose all strings have length $< \frac{\log|S|}{n} - 1$. Then all strings have length $\leq L = \left\lfloor \frac{\log|S|}{n} - 1 \right\rfloor$. But then $|S| \leq \left(2^{L+1}-1\right)^n < \left(2^{L+1}\right)^n \leq \left(2^{\frac{\log|S|}{n}}\right)^n = |S|$ which is a contradiction. ◀

▶ **Theorem 2.** *There exists a function $f(n) \in \Omega(n)$ such that if node labels have size $\leq f(n)$ bits, then there exists an unweighted graph with n nodes for which any routing scheme with stretch < 3 contains a routing program of size $\Omega(n)$ bits.*

Proof. Let $n = 2q + 2$ (the case where n is odd is analogous), $V = A \cup B$ where $A = \{a_0, a_1, ..., a_q\}$ and $B = \{b_0, b_1, ..., b_q\}$. Define $\mathcal{G}$ to be the set of all bipartite unweighted graphs with parts A and B in which a_0 has an edge to all vertices in B and b_0 has an edge to every vertex in A. The purpose of nodes a_0 and b_0 is to keep all graphs in $\mathcal{G}$ connected. Bipartiteness implies that all graphs in $\mathcal{G}$ have girth 4 and $|\mathcal{G}| = 2^{q^2}$.

We will now bound the number of graphs in $\mathcal{G}$ that a single routing scheme can support. This will give a lower bound on the number of different routing schemes for $\mathcal{G}$.

Fix any routing scheme $\mathcal{R}$ and node labels $\mathcal{L}$. For any node $v \in A \setminus \{a_0\}$, consider the map $X : B \to [\deg(v)]$ defined by $X(u) = \big(R_v(l_u, 0)\big)_2$, i.e., the port number at v through which a message originating from v to u is sent. Note that if $\{v, u\} \in E$ then the port $X(u)$ at v should lead to u; otherwise, the path taken will have length ≥ 3 violating the stretch requirement. We will now use this constraint to limit the number of different graphs in $\mathcal{G}$ that can be supported by $\mathcal{R}$ and $\mathcal{L}$ assuming the port assignments $\mathcal{T}$ can be varied.

Define $M = \max\{X(u) : u \in B\}$. First, observe that $\deg(v) \geq M$ because if $\deg(v) < M$, then routing to u with $X(u) = M$ will fail. Also observe that for each $1 \leq i \leq \deg(v)$, routing to the node behind port i will necessarily send the message directly to that node: $X(\tau_v(i)) = i$. Otherwise, routing to that node will traverse a different node and violate the stretch requirement. Hence, $M = \deg(v)$ and $X(u)$ partitions all nodes in B based on the outgoing port number, each partition being non-empty.

For all $1 \leq i \leq M$, define s_i to be the size of i's partition: $s_i = \big|\big\{u \in B : X(u) = i\big\}\big|$. We know that $\sum_{i=1}^{M} s_i = |B| = q + 1$ and that $\tau_v(i)$ has s_i possible values. Therefore, the neighbors of v can take at most $\prod_{i=1}^{M} s_i$ possible values, or in other words, $\mathcal{N}(v)$ is one of at most $\prod_{i=1}^{M} s_i$ possible configurations. It is known that the geometric average of a set of non-negative numbers cannot exceed their arithmetic average. Hence, $\prod_{i=1}^{M} s_i \leq \big(\frac{q+1}{M}\big)^M$. Differentiating the logarithm with respect to M, we find that the value is maximized when $M = \frac{q+1}{e}$ and thus $\prod_{i=1}^{M} s_i \leq e^{\frac{q+1}{e}}$. Hence, the routing scheme $\mathcal{R}$ with labels $\mathcal{L}$ satisfies at most $e^{\frac{q+1}{e} \cdot q}$ graphs in $\mathcal{G}$. I.e., if for a graph G, routing scheme $\mathcal{R}$, labels $\mathcal{L}$, and port assignment $\mathcal{T}$ we define the predicate $\mathrm{Sat}(G, \mathcal{R}, \mathcal{L}, \mathcal{T})$ to be true if and only if $(\mathcal{R}, \mathcal{L}, \mathcal{T})$ satisfy G with routing stretch < 3, then

$$\Big|\big\{G \in \mathcal{G} : \exists \mathcal{T}, \mathrm{Sat}(G, \mathcal{R}, \mathcal{L}, \mathcal{T})\big\}\Big| \leq e^{\frac{q+1}{e} \cdot q}.$$

Since the labels have size at most $f(n)$ bits, the number of possible configurations of labels $\mathcal{L}$ is at most $(2^{f(n)+1} - 1)^n < 2^{(f(n)+1)n}$. Hence, the number of graphs in $\mathcal{G}$ that $\mathcal{R}$ can satisfy is

$$\Big|\big\{G \in \mathcal{G} : \exists (\mathcal{T}, \mathcal{L}), \mathrm{Sat}(G, \mathcal{R}, \mathcal{L}, \mathcal{T})\big\}\Big| < e^{\frac{q+1}{e} \cdot q} \cdot 2^{(f(n)+1)n}.$$

Therefore, the number of routing schemes necessary to satisfy each graph in $\mathcal{G}$ is larger than

$$\frac{|\mathcal{G}|}{e^{\frac{q+1}{e} \cdot q} \cdot 2^{(f(n)+1)n}} = \frac{2^{q^2}}{e^{\frac{q+1}{e} \cdot q} \cdot 2^{(f(n)+1)n}}.$$

Since

$$\log \frac{2^{q^2}}{e^{\frac{q+1}{e}\cdot q}\cdot 2^{(f(n)+1)n}} = q^2 - \frac{\log e}{e}q(q+1) - \big(f(n)+1\big)n =$$
$$\Big(\frac{n}{2}-1\Big)^2 - \frac{\log e}{2e}n\Big(\frac{n}{2}-1\Big) - \big(f(n)+1\big)n =$$
$$\Big(\frac{1}{4} - \frac{\log e}{4e}\Big)n^2 - \big(f(n)+1\big)n + \Big(\frac{\log e}{2e} - \frac{1}{2}\Big)n + 1 \in \Omega(n^2)$$

for an appropriate $f(n)$, one of the graphs in $\mathcal{G}$ requires $\Omega(n)$ bits of state at some node by lemma 1. ◀

A small modification of this argument lets us prove a slightly stronger statement, that any constant fraction of n nodes must have $\Omega(n)$ bits of state on some graph. Using the same family of graphs $\mathcal{G}$, simple counting manipulations, and the bound on the number of possible neighbor sets of a given node from above, we count the number of possible graphs possessing a routing scheme with too many small routing programs. We find that this number is less than $|\mathcal{G}|$ implying that at least one graph must not have too many small routing programs.

▶ **Theorem 3.** *For any $0 < c < 1$ there exist functions $f(n), g(n) \in \Omega(n)$ (f depends on c) such that if node labels have size $\leq f(n)$ bits, then there exists an unweighted graph with n nodes for which any routing scheme with stretch < 3 contains $\geq cn$ routing programs of size $\geq g(n)$ bits.*

Proof. Let $0 < c < 1$, $f(n) = \frac{1-c}{40}n$, $g(n) = \frac{n}{8}$, and define $k = \lfloor \frac{1-c}{2}n - 1 \rfloor$. We consider the same set of graphs $\mathcal{G}$ as in the previous theorem.

Fix node labels $\mathcal{L}$ and let $A' \subseteq A \setminus \{a_0\}$ such that $|A'| = k$. Now suppose for all $v \in A'$, $|R_v| < g(n)$, v's routing program is smaller than $g(n)$ bits. Then there are less than $2^{g(n)+1}$ different routing programs for each node in A'. By the argument in the previous theorem, each node in A' can have $< e^{\frac{q+1}{e}} \cdot 2^{g(n)+1}$ distinct sets of neighbors. Trivially, all other nodes in A can have at most 2^q distinct sets of neighbors. Therefore, the number of possible graphs is

$$\Big|\Big\{G \in \mathcal{G} : \exists(\mathcal{T},\mathcal{R}), \big(\forall v \in A', |R_v| < g(n)\big) \wedge \mathrm{Sat}(G,\mathcal{R},\mathcal{L},\mathcal{T})\Big\}\Big| <$$
$$\big(e^{\frac{q+1}{e}} \cdot 2^{g(n)+1}\big)^k \cdot \big(2^q\big)^{q-k}.$$

Therefore, if A' can be arbitrary, the number of possible graphs is

$$\Big|\Big\{G \in \mathcal{G} : \exists(\mathcal{T},\mathcal{R}), \mathrm{Sat}(G,\mathcal{R},\mathcal{L},\mathcal{T}) \wedge \big|\{v \in A \setminus \{a_0\} : |R_v| < g(n)\}\big| \geq k\Big\}\Big| =$$
$$\Big|\Big\{G \in \mathcal{G} : \exists(\mathcal{T},\mathcal{R},A'), \big(\forall v \in A', |R_v| < g(n)\big) \wedge \mathrm{Sat}(G,\mathcal{R},\mathcal{L},\mathcal{T})\Big\}\Big| <$$
$$\big(e^{\frac{q+1}{e}} \cdot 2^{g(n)+1}\big)^k \cdot \big(2^q\big)^{q-k} \cdot 2^q$$

(assuming $\exists A'$ means A' is taken from $A \setminus \{a_0\}$ such that $|A'| = k$).

If, in addition, the node labels can be varied, the number of possible graphs is

$$\Big|\Big\{G \in \mathcal{G} : \exists(\mathcal{T},\mathcal{R},\mathcal{L}), \mathrm{Sat}(G,\mathcal{R},\mathcal{L},\mathcal{T}) \wedge \big|\{v \in A \setminus \{a_0\} : |R_v| < g(n)\}\big| \geq k\Big\}\Big| <$$
$$\big(e^{\frac{q+1}{e}} \cdot 2^{g(n)+1}\big)^k \cdot \big(2^q\big)^{q-k} \cdot 2^q \cdot 2^{(f(n)+1)n}.$$

This is less than $2^{q^2}/2$ since

$$\log\left(\left(e^{\frac{q+1}{e}}\cdot 2^{g(n)+1}\right)^k\cdot(2^q)^{q-k}\cdot 2^q\cdot 2^{(f(n)+1)n}\right)=$$
$$k\left(\frac{q+1}{e}\cdot\log e+g(n)+1\right)+q(q-k)+q+(f(n)+1)n=$$
$$-k\left(q-\frac{\log e}{e}(q+1)-g(n)-1\right)+q^2+q+nf(n)+n=$$
$$-\left\lfloor\frac{1-c}{2}n-1\right\rfloor\left(\frac{n}{2}-1-\frac{\log e}{2e}n-\frac{n}{8}-1\right)+\frac{n}{2}\left(\frac{n}{2}-1\right)+\frac{1-c}{40}n^2+n<$$
$$-\left(\frac{1-c}{2}n-2\right)\left(\frac{n}{10}-2\right)+\frac{n}{2}\left(\frac{n}{2}-1\right)+\frac{1-c}{40}n^2+n=$$
$$\left(\frac{1}{4}-\frac{1-c}{40}\right)n^2+O(n)<\left(\frac{n}{2}-1\right)^2-1=q^2-1.$$

By symmetry, the number of possible graphs with a routing scheme having many small programs in $B\setminus\{b_0\}$

$$\left|\left\{G\in\mathcal{G}:\exists(\mathcal{T},\mathcal{R},\mathcal{L}),\mathrm{Sat}(G,\mathcal{R},\mathcal{L},\mathcal{T})\wedge\left|\left\{v\in B\setminus\{b_0\}:|R_v|<g(n)\right\}\right|\geq k\right\}\right|<2^{q^2}/2.$$

Hence, there are less than 2^{q^2} graphs which have a routing scheme with at least k small programs in $A\setminus\{a_0\}$ or $B\setminus\{b_0\}$:

$$\Big|\Big\{G\in\mathcal{G}:\exists(\mathcal{T},\mathcal{R},\mathcal{L}),\mathrm{Sat}(G,\mathcal{R},\mathcal{L},\mathcal{T})\wedge$$
$$\left(|\{v\in A\setminus\{a_0\}:|R_v|<g(n)\}|\geq k\vee|\{v\in B\setminus\{b_0\}:|R_v|<g(n)\}|\geq k\right)\Big\}\Big|<2^{q^2}.$$

So at least one graph G must have $<k$ small routing programs in both parts of the graph (excluding special vertices a_0 and b_0):

$$\forall(\mathcal{T},\mathcal{R},\mathcal{L}),\mathrm{Sat}(G,\mathcal{R},\mathcal{L},\mathcal{T})\rightarrow$$
$$\left|\left\{v\in A\setminus\{a_0\}:|R_v|<g(n)\right\}\right|<k\wedge\left|\left\{v\in B\setminus\{b_0\}:|R_v|<g(n)\right\}\right|<k.$$

It follows that the number of large programs must exceed $n-2-2k\geq cn$. ◀

We note that it's not true that *all* nodes must store $\Omega(n)$ bits of information. One can choose a node v and include in the label for each node u v's port toward u. This adds $O(\log n)$ bits to each node label and v can have constant size state.

3 Lower bounds with adversarial port numbers

3.1 Stretch < 3

We will now prove an obvious fact which we haven't seen elsewhere: if the port assignment $\mathcal{T}$ are adversarially chosen and node labels are not, then some node must have $\Omega(n\log n)$ bits of state. Note the additional $\log n$ factor compared to the previous theorems. This implies that the trivial shortest paths routing scheme has optimal space up to a constant. Our main theorem in this section is strictly more general; however, this smaller result serves as a warmup and also lets us prove an $\Omega(n\log n)$ bit space lower bound for any constant fraction of nodes, similar to theorem 3.

▶ **Theorem 4.** *There exists a function $f(n)\in\Omega(n\log n)$ such that if node labels have size $\leq f(n)$ bits, then there exists an unweighted graph with n nodes for which any routing scheme with adversarial port assignment and stretch <3 contains a routing program of size $\Omega(n\log n)$ bits.*

Proof. Let $n = 2q$ (for odd n the proof is similar) and define $f(n) = \frac{\log(q!)}{2} \in \Omega(n \log n)$. We consider only one graph $G = K_{q,q}$, a full bipartite graph with parts of size q, and vary the port assignments $\mathcal{T}$. Let $\tilde{\mathcal{T}}$ be the set of all possible port assignments for G. Then $|\tilde{\mathcal{T}}| = (q!)^n$.

Because the stretch must be less than 3, routing from a node u in one part of G to a node v in the other part must traverse the edge $\{u, v\}$. Thus if we fix the node labels $\mathcal{L}$, every port assignment $\mathcal{T}$ requires a distinct routing scheme $\mathcal{R}$. Then if node labels $\mathcal{L}$ are varied, same routing scheme $\mathcal{R}$ can support at most $2^{(f(n)+1)n}$ port assignments:

$$\left|\left\{\mathcal{T} \in \tilde{\mathcal{T}} : \exists \mathcal{L}, \mathrm{Sat}(G, \mathcal{R}, \mathcal{L}, \mathcal{T})\right\}\right| \leq 2^{(f(n)+1)n}.$$

Then there must be at least

$$\frac{|\tilde{\mathcal{T}}|}{2^{(f(n)+1)n}} = \frac{(q!)^n}{2^{(\frac{\log(q!)}{2}+1)n}}$$

distinct routing schemes to support all possible port assignments.

Since

$$\log \frac{(q!)^n}{2^{(\frac{\log(q!)}{2}+1)n}} = n \log(q!) - \left(\frac{\log(q!)}{2} + 1\right) n \in \Omega(n^2 \log n),$$

some routing program must have size $\Omega(n \log n)$ bits. ◀

Using the techniques in the proof of theorem 3, one can prove this space lower bound for any constant fraction of nodes.

▶ **Theorem 5.** *For any $0 < c < 1$ there exist functions $f(n), g(n) \in \Omega(n \log n)$ (f depends on c) such that if node labels have size $\leq f(n)$ bits, then there exists an unweighted graph with n nodes for which any routing scheme with adversarial port assignment and stretch < 3 contains $\geq cn$ routing programs of size $\geq g(n)$ bits.*

3.2 Stretch $< 2k + 1$

We would like to show for any integer $k \geq 1$ that when port assignment is adversarial, the worst case per-node space complexity of a routing scheme of stretch $< 2k + 1$ is $\Omega\left(n^{\frac{1}{k}} \log n\right)$. We are able to show this only assuming a well-known girth conjecture by Erdős.

▶ **Definition 6.** *We say that an unweighted undirected graph G has girth s if and only if all cycles in G have length $\geq s$.*

▶ **Definition 7.** *For any integers $s, n \geq 1$, define $M_s(n)$ to be the largest m such that there exists a graph with n vertices, m edges and girth s.*

It is known that any graph with n nodes and $n^{1+\frac{1}{k}}$ edges must have a cycle of length at most $2k$. So, $M_{2k+2}(n) \in O\left(n^{1+\frac{1}{k}}\right)$. However, $M_{2k+2}(n) \in \Omega\left(n^{1+\frac{1}{k}}\right)$ has only been proven for $k = 1, 2, 3, 5$; for other k it remains an open question. Also, $M_{2k+1}(n) \in \Theta\left(M_{2k+2}(n)\right)$ since any graph with n vertices, m edges and girth $2k + 1$ must contain a bipartite subgraph with $\geq \frac{m}{2}$ edges (which necessarily has girth $2k + 2$). For an overview of the best known lower bounds for $M_s(n)$, refer to [16]. We note that most constructions of large graphs with some fixed girth rely on finite fields, and the number of vertices n is thus required to be some polynomial of a prime or a prime power. To prove a lower bound for all sufficiently large n, and thus justify the use of Ω, one can employ Bertrand's postulate which states that for all

natural numbers i, $p_{i+1} < 2p_i$, where p_i is the i-th prime number. Justification of Ω easily follows from here: given an integer n, we construct a dense large girth graph with p_i vertices where $\frac{n}{2} < p_i \leq n$, and add missing $n - p_i$ vertices along with $n - p_i$ edges keeping the final graph connected.

Having a graph G with n vertices and $\Omega\big(n^{1+\frac{1}{k}}\big)$ edges lets us show that there must be at least $2^{\Omega(n^{1+\frac{1}{k}} \log n)}$ routing schemes to satisfy all possible node labels $\mathcal{T}$ for G. Hence, at least some node must use $\Omega\big(n^{\frac{1}{k}} \log n\big)$ bits of space in the worst case. Weaker lower bounds for $M_{2k+2}(n)$ give corresponding weaker results.

▶ **Theorem 8.** *Let $k \geq 1$ be an integer constant. If port numbers are adversarially chosen, $\frac{m}{n}$ is sufficiently large, there exists a graph with n vertices, m edges and girth $2k+2$, and all node labels have size $\leq L = \frac{1}{16k} \cdot \log \frac{m}{n} \cdot \frac{m}{n} - 1$, then there also exists a port assignment $\mathcal{T}$ for G such that any routing scheme for $(G, \mathcal{T})$ of stretch $< 2k+1$ requires $\frac{1}{32k} \cdot \log \frac{m}{n} \cdot \frac{m}{n}$ bits of space at some node.*

Proof. Let $G = (V, E)$ be a graph with n vertices, m edges and girth $2k+2$, where $V = [n]$. For any pair of neighbors $\{x, y\} \in E$, the route from x to y chosen by a routing scheme of stretch $< 2k+1$ must be of length $\leq 2k$. Therefore, during routing from x to y, x must send the packet to y, most likely as the last step; otherwise, there would be a cycle in G of length $\leq 2k+1$ which violates the girth requirement. This doesn't, however, mean that routing from x to y must take the shortest path. x might send the message to some other neighbor z which returns the message to x (with a modified header) which now sends the message to y. In general, if you trace the message from x to y, the visited nodes and traversed edges form an arbitrary tree (see figure 1).

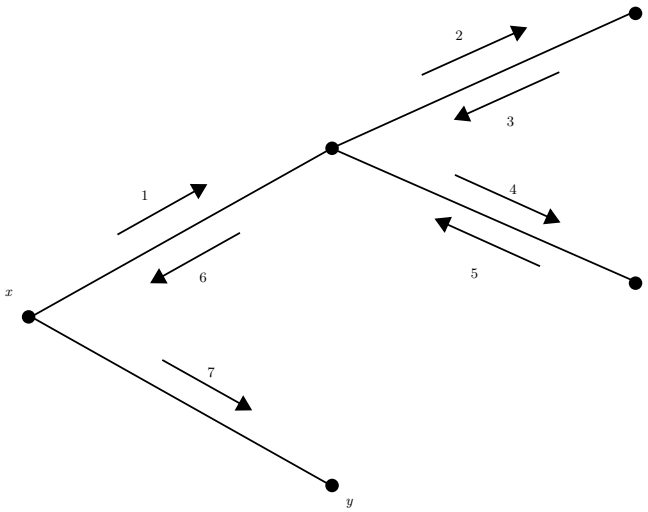

Figure 1 Routing with stretch $< 2k+1$ in a graph with girth $2k+2$.

We consider only a single graph G and all possible port assignments $\mathcal{T}$ for G. Define $m_i = \deg(i)$. Then the number of all possible port assignments for G is $\prod_{i=1}^{n} m_i!$. As in theorem 2, we will argue that any routing scheme cannot satisfy many distinct port assignments.

Fix the node labels $\mathcal{L}$ (as the proof of theorem 2 shows, they do not have much of an effect) and a routing scheme $\mathcal{R}$. Here is the main idea of the proof. Take any pair of neighbors $\{x, y\} \in E$. Then the port $p = R_x(y, 0)_2$ through which x will send the message toward y is

determined. The neighbor behind port p can be arbitrary, but if we fix it, say z, and we fix z's port to x, then the next hop is determined and the next out port through which z will forward the message will also be determined. We can keep tracing the packet through the network fixing the ports along the way. Eventually, however, node x must send the packet to node y, and this is where we can "extract" a port assignment from the routing programs. Hence, the routing scheme must "store" approximately every $4k$-th port assignment.

To formalize this argument, we describe a program that, given node labels $\mathcal{L}$, a routing scheme $\mathcal{R}$, port assignment on vertices with small degrees $S = \left\{x \in V : m_x < \frac{m}{n}\right\}$, and a bit string, can recover the port assignment $\mathcal{T}$ on all vertices. The program will iteratively pick neighbor vertices x and y such that the port at x toward y is undefined, and try to simulate the routing as in the example above reading the number of hops until the edge $\{x, y\}$ is traversed and unknown port assignments along the path from the bit string. We will then argue that this bit string does not need to be long, which intuitively implies that the routing scheme $\mathcal{R}$ must contain much information.

The program will work with a list of partial functions, one for each vertex, signifying which ports have already been fixed. Denote the set of all possible such lists by $\mathcal{P}$, and define $\vec{P} \in \mathcal{P}$ if and only if $\vec{P} = (P_1, P_2, ..., P_n)$ where each $P_i \subseteq [m_i] \times \mathcal{N}(i)$ such that for each $p \in [m_i]$ there is at most one j such that $(p, j) \in P_i$ and for each j there is at most one p such that $(p, j) \in P_i$.

We now formally describe the program which has access to $\mathcal{L} = \{l_i\}_{i\in[n]}$, $\mathcal{R} = \{R_i\}_{i\in[n]}$ and $\vec{P}$ such that $\forall i \in S, |\vec{P}_i| = m_i$ and $\forall i \in V \setminus S, |\vec{P}_i| = 0$, and reads bits from the input bit string.

while $\exists x \in [n], |\vec{P}_x| < m_x$ **do**
  $X \leftarrow \{i \in [n] : |\vec{P}_i| < m_i\}$
  $M \leftarrow \max\{m_i : i \in X\}$
  $x \leftarrow \min\{i \in X : m_i = M\}$
  $y \leftarrow \min\{i \in \mathcal{N}(x) : (\vec{P}_x)^{-1}(i) \text{ is undefined}\}$
  $r \leftarrow$ read $\lceil \log(2k-1) \rceil$-bit number ▷ read the number of hops
  $h \leftarrow l_y$
  $p_{\text{in}} \leftarrow 0$
  for r times **do**
    $p_{\text{out}} \leftarrow R_x(h, p_{\text{in}})_2$
    $h \leftarrow R_x(h, p_{\text{in}})_1$
    if $\vec{P}_x(p_{\text{out}})$ is undefined **then**
      $i \leftarrow$ read $\lceil \log m_x \rceil$-bit number ▷ read the "ID" of the neighbor behind port p_{out}
      $\vec{P}_x \leftarrow \vec{P}_x \cup \{(p_{\text{out}}, \mathcal{N}(x)(i))\}$ ▷ update the known port assignments
    end if
    $z \leftarrow \vec{P}_x(p_{\text{out}})$ ▷ next visited node
    if $(\vec{P}_z)^{-1}(x)$ is undefined **then**
      $p \leftarrow$ read $\lceil \log m_z \rceil$-bit number ▷ read the port we are entering through at z
      $\vec{P}_z \leftarrow \vec{P}_z \cup \{(p, x)\}$
    end if
    $p_{\text{in}} \leftarrow (\vec{P}_z)^{-1}(x)$
    $x \leftarrow z$
  end for
  $p_{\text{out}} \leftarrow R_x(h, p_{\text{in}})_2$
  $\vec{P}_x \leftarrow \vec{P}_x \cup (p_{\text{out}}, y)$ ▷ learn this port assignment "for free"
end while
return $\vec{P}$

We will now bound the length of the bit string needed for this program to run correctly. The following outer loop invariant is claimed: if b is the number of bits read so far, then

$$b \leq \sum_{i \in V \setminus S} \left|\vec{P}_i\right| \cdot \Big(\lceil \log(2k) \rceil + \big(1 - \frac{1}{4k}\big) \lceil \log m_i \rceil \Big).$$

It is correct before the loop starts since $b = 0$. Assume it is correct after some number of iterations, and let b' be the number of bits read and $\vec{P}'$ be the updated port assignment at the end of the iteration. We need to prove that

$$b' \leq \sum_{i \in V \setminus S} \left|\vec{P}'_i\right| \cdot \Big(\lceil \log(2k) \rceil + \big(1 - \frac{1}{4k}\big) \lceil \log m_i \rceil \Big).$$

Let t be the number of individual port assignments discovered through reading from the bit string, and let $z_1, ..., z_t$ be the nodes that own those ports. Notice that $t \leq 4k - 2$ since after $4k - 2$ traversed ports, the next port must lead from x directly to y and we do not read it from the bit string. Also notice that the degrees of the nodes $z_1, ..., z_t$ do not exceed that of x, the source node. This is because the algorithm picks x with the largest degree among those that have free ports. Then

$$\sum_{i \in V \setminus S} \left|\vec{P}'_i\right| \cdot \Big(\lceil \log(2k) \rceil + \big(1 - \frac{1}{4k}\big) \lceil \log m_i \rceil \Big) =$$

$$\sum_{i \in V \setminus S} \left|\vec{P}_i\right| \cdot \Big(\lceil \log(2k) \rceil + \big(1 - \frac{1}{4k}\big) \lceil \log m_i \rceil \Big) + \sum_{i=1}^{t} \Big(\lceil \log(2k) \rceil + \big(1 - \frac{1}{4k}\big) \lceil \log m_{z_i} \rceil \Big) +$$

$$\Big(\lceil \log(2k) \rceil + \big(1 - \frac{1}{4k}\big) \lceil \log m_x \rceil \Big) \geq$$

$$b + \lceil \log(2k) \rceil + \big(1 - \frac{1}{4k}\big) \Big(\sum_{i=1}^{t} \lceil \log m_{z_i} \rceil + \lceil \log m_x \rceil \Big) =$$

$$b + \lceil \log(2k) \rceil + \sum_{i=1}^{t} \lceil \log m_{z_i} \rceil + \lceil \log m_x \rceil - \frac{1}{4k} \Big(\sum_{i=1}^{t} \lceil \log m_{z_i} \rceil + \lceil \log m_x \rceil \Big) [\geq]$$

Since in this iteration of the outer loop we read $\lceil \log(2k-1) \rceil + \sum_{i=1}^{t} \lceil \log m_{z_i} \rceil$ bits,

$$[\geq] b' + \lceil \log m_x \rceil - \frac{1}{4k} \Big(\sum_{i=1}^{t} \lceil \log m_{z_i} \rceil + \lceil \log m_x \rceil \Big) \geq$$

$$b' + \lceil \log m_x \rceil - \frac{1}{4k} \big((4k-2) \lceil \log m_x \rceil + \lceil \log m_x \rceil \big) \geq b'.$$

Hence, the loop invariant holds and the algorithm reads at most a total of B bits where

$$B = \sum_{i \in V \setminus S} m_i \big(\lceil \log(2k) \rceil + (1 - \frac{1}{4k}) \lceil \log m_i \rceil \big) \leq \sum_{i \in V \setminus S} m_i \big((1 - \frac{1}{4k}) \log m_i + \log k + 3 \big).$$

Since node labels have size $\leq L = \frac{1}{16k} \cdot \log \frac{m}{n} \cdot \frac{m}{n} - 1$, the number of all possible $\mathcal{L}$ is $\big(2^{L+1} - 1\big)^n \leq 2^{(L+1)n} = 2^{\frac{1}{16k} \cdot \log \frac{m}{n} \cdot m}$. Then, defining $\mathrm{Sat}(G, \mathcal{R}, \mathcal{L}, \mathcal{T})$ true if and only if $\mathcal{R}$ is a correct routing scheme with stretch $< 2k + 1$ for graph G with labels $\mathcal{L}$ and port assignment $\mathcal{T}$,

$$\Big| \big\{ \mathcal{T} : \exists \mathcal{L}, \mathrm{Sat}(G, \mathcal{R}, \mathcal{L}, \mathcal{T}) \big\} \Big| \leq \qquad\qquad 2^{\frac{1}{16k} \cdot \log \frac{m}{n} \cdot m} \cdot \prod_{i \in S} m_i! \cdot 2^B.$$

Therefore, the number of routing schemes necessary to satisfy all port assignments is at least

$$\frac{\prod_{i=1}^{n} m_i!}{2^{\frac{1}{16k}\cdot\log\frac{m}{n}\cdot m}\cdot\prod_{i\in S} m_i!\cdot 2^B}.$$

We now calculate the logarithm of this expression.

$$\log\frac{\prod_{i=1}^{n} m_i!}{2^{\frac{1}{16k}\cdot\log\frac{m}{n}\cdot m}\cdot\prod_{i\in S} m_i!\cdot 2^B} = \log\frac{\prod_{i\in V\setminus S} m_i!}{2^{\frac{1}{16k}\cdot\log\frac{m}{n}\cdot m}\cdot 2^B} =$$

$$\sum_{i\in V\setminus S}\log\left(m_i!\right) - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m - B[\geq]$$

Using Stirling's approximation,

$$[\geq]\sum_{i\in V\setminus S}(m_i\log m_i - cm_i) - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m - B =$$

$$\sum_{i\in V\setminus S}(m_i\log m_i - cm_i) - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m - \sum_{i\in V\setminus S} m_i\Big(\big(1-\frac{1}{4k}\big)\log m_i + \log k + 3\Big) =$$

$$\sum_{i\in V\setminus S} m_i\Big(\log m_i - c - \big(1-\frac{1}{4k}\big)\log m_i - \log k - 3\Big) - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m =$$

$$\sum_{i\in V\setminus S} m_i\big(\frac{1}{4k}\log m_i - c - \log k - 3\big) - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m[\geq]$$

If $i\in V\setminus S$, then $m_i\geq\frac{m}{n}$. Thus,

$$[\geq]\sum_{i\in V\setminus S} m_i\big(\frac{1}{4k}\log\frac{m}{n} - c - \log k - 3\big) - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m[\geq]$$

Assuming $\frac{m}{n}$ is sufficiently large,

$$[\geq]\sum_{i\in V\setminus S}\frac{1}{8k}m_i\log\frac{m}{n} - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m =$$

$$\frac{1}{8k}\cdot\log\frac{m}{n}\cdot\sum_{i\in V\setminus S} m_i - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m =$$

$$\frac{1}{8k}\cdot\log\frac{m}{n}\cdot\Big(2m - \sum_{i\in S} m_i\Big) - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m \geq$$

$$\frac{1}{8k}\cdot\log\frac{m}{n}\cdot\Big(2m - n\cdot\frac{m}{n}\Big) - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m \geq$$

$$\frac{1}{8k}\cdot\log\frac{m}{n}\cdot m - \frac{1}{16k}\cdot\log\frac{m}{n}\cdot m =$$

$$\frac{1}{16k}\cdot\log\frac{m}{n}\cdot m.$$

By lemma 1, there is a port assignment on which some node must use $\frac{1}{16k}\cdot\log\frac{m}{n}\cdot\frac{m}{n} - 1 \geq \frac{1}{32k}\cdot\log\frac{m}{n}\cdot\frac{m}{n}$ bits of space. ◀

▶ **Corollary 9.** *Let $k\geq 1$ be an integer constant. If $M_{2k+2}(n)\in\Omega(n^{1+\frac{1}{k}})$ and node labels have size $\leq\Omega(n^{\frac{1}{k}}\log n)$, then for any n there is a graph G on n nodes and a port assignment $\mathcal{T}$ for G such that any routing scheme for $(G,\mathcal{T})$ of stretch $<2k+1$ requires $\Omega(n^{\frac{1}{k}}\log n)$ bits of space at some node.*

4 Known techniques require girth conjecture

All known proofs of lower bounds for space complexity of non-shortest path routing schemes in the labeled setting, such as the ones in [14], [10] or this paper, consist of proving that a large number of routing schemes are necessary to satisfy all possible graphs and port assignments (if adversarially chosen). Then it follows that some graph and port assignment requires a large number of bits stored at some node.

We note that proving that a large number of routing schemes is necessary must require existence of graphs with large girth and many edges. Therefore, if one wishes to not use a girth conjecture, some other technique, such as the one in [2] for the name-independent setting, is needed.

We utilize spanners with low stretch and large girth.

▶ **Definition 10.** *Let G be a graph with n vertices. We call a subgraph S of G a t-spanner if and only if for any two vertices u, v connected in G, $d_S(u,v) \leq t \cdot d_G(u,v)$ (distance in S is stretched at most t times compared to the distance in G).*

The following lemma is proved in [6], but we provide the proof for completeness.

▶ **Lemma 11.** *Let G be a graph with n vertices. There exists a t-spanner S of G with girth $t+2$.*

Proof. We construct a spanner S similarly to Kruskal's minimum spanning tree algorithm. Go through all edges in G, and for each edge $\{u,v\}$, add it to S if and only if there isn't a path between u and v in S of length $\leq t$.

Clearly, at the end of the algorithm, for any edge $\{u,v\}$ in G, $d_S(u,v) \leq t$. Therefore, S is a t-spanner.

We only need to show that S has girth $t+2$. Suppose during the algorithm we add an edge and create a cycle of length $l \leq t+1$. Then there was already a path between the endpoints of length $l-1 \leq t$, and we shouldn't add an edge. ◀

▶ **Theorem 12.** *Suppose $n \geq 2$ and $s \geq 2$ are integers, and N routing schemes are necessary to satisfy all graphs with n vertices and all port assignments with stretch $< s$. Then $M_{s+1}(n) \geq \frac{\log N}{4 \log n} - 1$; i.e., there exists a graph with $\geq \frac{\log N}{4 \log n} - 1$ edges and girth $s+1$.*

Proof. Suppose $M_{s+1}(n) < \frac{\log N}{4 \log n} - 1$; i.e., all graphs with girth $s+1$ have $< \frac{\log N}{4 \log n} - 1$ edges. We want to show that then less than N routing schemes are sufficient to satisfy all graphs and port assignments.

Given a graph G and a port assignment $\mathcal{T}$, we simply encode in each node's routing program its ID, an $(s-1)$-spanner S of G with girth $s+1$ which exists by lemma 11, and the port assignment $\mathcal{T}$. All this information allows for stretch $s-1$ routing.

It is easy to see that there are at most $(n^2)^m$ graphs with n vertices and m edges. Also, for each such graph there are at most $(n^2)^m$ port assignments. Therefore, the number of such routing schemes is at most

$$\sum_{i=0}^{M_{s+1}(n)} \left(n^2\right)^i \cdot \left(n^2\right)^i = \sum_{i=0}^{M_{s+1}(n)} \left(n^4\right)^i = \frac{\left(n^4\right)^{M_{s+1}(n)+1} - 1}{n^4 - 1} < \left(n^4\right)^{M_{s+1}(n)+1}.$$

Since $\log\left(\left(n^4\right)^{M_{s+1}(n)+1}\right) = 4\big(M_{s+1}(n)+1\big)\log n < 4\Big(\frac{\log N}{4 \log n}\Big)\log n = \log N$, less than N routing schemes can satisfy all graphs with n vertices and all port assignments with stretch $< s$. This is a contradiction. ◀

▶ **Corollary 13.** *Let $k \geq 1$ be an integer constant. If $2^{\Omega(n^{1+\frac{1}{k}} \log n)}$ routing schemes are necessary to satisfy all graphs with n vertices and all port assignments with stretch $< 2k+1$, then $M_{2k+2} \in \Omega\left(n^{1+\frac{1}{k}}\right)$.*

References

1 Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Routing with improved communication-space trade-off. In Rachid Guerraoui, editor, *Distributed Computing*, pages 305–319, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. doi:10.1007/978-3-540-30186-8_22.

2 Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. On space-stretch trade-offs: Lower bounds. In *Proceedings of the Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '06, pages 207–216, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1148109.1148143.

3 Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. On space-stretch trade-offs: Upper bounds. In *Proceedings of the Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '06, pages 217–224, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1148109.1148144.

4 Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. *ACM Trans. Algorithms*, 4(3), July 2008. doi:10.1145/1367064.1367077.

5 Ittai Abraham and Dahlia Malkhi. Name independent routing for growth bounded networks. In *Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '05, pages 49–55, New York, NY, USA, 2005. Association for Computing Machinery. doi:10.1145/1073970.1073978.

6 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete Comput. Geom.*, 9(1):81–100, December 1993. doi:10.1007/BF02189308.

7 Harry Buhrman, Jaap-Henk Hoepman, and Paul Vitányi. Optimal routing tables. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '96, pages 134–142, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/248052.248076.

8 Shiri Chechik. Compact routing schemes with improved stretch. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13, pages 33–41, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2484239.2484268.

9 Cyril Gavoille and Marc Gengler. Space-efficiency for routing schemes of stretch factor three. Technical report, Laboratoire Bordelais de Recherche en Informatique, Université Bordeaux, 1997. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.5857&rep=rep1&type=pdf.

10 Cyril Gavoille and Marc Gengler. Space-efficiency for routing schemes of stretch factor three. *Journal of Parallel and Distributed Computing*, 61(5):679–687, 2001. doi:10.1006/jpdc.2000.1705.

11 Cyril Gavoille and Stéphane Pérennès. Memory requirement for routing in distributed networks. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '96, pages 125–133, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/248052.248075.

12 Goran Konjevod, Andréa W. Richa, and Donglin Xia. Scale-free compact routing schemes in networks of low doubling dimension. *ACM Trans. Algorithms*, 12(3), June 2016. doi:10.1145/2876055.

13 J. Matousek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93:333–344, 1996. doi:10.1007/BF02761110.

14 David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, July 1989. doi:10.1145/65950.65953.

15 Liam Roditty and Roei Tov. *New Routing Techniques and Their Applications*, pages 23–32. Association for Computing Machinery, New York, NY, USA, 2015. doi:10.1145/2767386.2767409.

16 Mikkel Thorup and Uri Zwick. Approximate distance oracles. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 183–192. ACM, 2001. doi:10.1145/380752.380798.

17 Mikkel Thorup and Uri Zwick. Compact routing schemes. In Arnold L. Rosenberg, editor, *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2001, Heraklion, Crete Island, Greece, July 4-6, 2001*, pages 1–10. ACM, 2001. doi:10.1145/378580.378581.