On Space-Stretch Trade-Offs: Lower Bounds

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

Cyril Gavoille*
Laboratoire Bordelais de Recherche en Informatique
University of Bordeaux
Bordeaux, France
gavoille@labri.fr

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

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 ratio between the cost of the route induced by the scheme and the cost of a minimum cost path between the same pair. 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 maximum stretch strictly less than $2k+1$ requires $\Omega((n \log n)^{1/k})$-bit routing tables. We extend our results to lower bound the average-stretch, showing that for any integer $k \geq 1$ any name-independent routing scheme with $(n/(9k))^{1/k}$-bit routing tables has average-stretch of at least $k/4 + 7/8$. This result is in sharp contrast to recent results on the average-stretch of labeled routing schemes.

Categories and Subject Descriptors: C.2.1 [Computer-Communication Networks]: Network Architecture and Design – *Distributed networks*; G.2.2 [Discrete Mathematics]: Graph Theory – *Network problems, Graph labeling.*

General Terms: Algorithms, Theory.

Keywords: Compact Routing.

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. Given a source s and a destination t let $stretch(s,t)$ be the ratio between the cost of routing from the source to the destination and the cost of a minimum cost path between s and t. Let

$$\text{max-stretch} = \max_{s \neq t} stretch(s,t)$$

$$\text{average-stretch} = \frac{1}{n(n-1)} \sum_{s \neq t} stretch(s,t)$$

The efficiency of a routing scheme is measured by its *stretch factor*, the max-stretch (or average-stretch)[1]. 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 problem, called *labeled routing*, asymptotically tight lower bounds and upper bounds are known for max-stretch (see [14]). 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

*Supported by the projects "PairAPair" and "GeoComp" of the ACI Masses de Données.

SPAA'06, July 30–August 2, 2006, Cambridge, Massachusetts, USA.

[1]When we speak of stretch without a prefix we mean max-stretch.

in the network, but we must do so without exceeding the distance to the target.

1.1 Our contributions

We first 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 less than $\Omega((n \log n)^{1/k})$ space routing tables must have networks in which the stretch is at least $2k+1$.

Theorem 1. *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 [13]. Or, as in Thorup and Zwick [14] obtain $\Omega(n^{1/k})$ but their proof relies on an unproven Erdös girth Conjecture [4]. Moreover, our lower bound has a subtle $(\log n)^{1/k}$ factor that was not obtained by [14]. 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 [9] is optimal up to polylogarithmic storage factors.

Our lower bound technique for name-independent routing can be applied to average-stretch.

Theorem 2. *For each integer $k \geq 1$ there is a weighted n-node star for which every name-independent routing scheme with less than $(n/(9k))^{1/k}$ bits of memory requirements has average-stretch at least $k/4 + 7/8$.*

This results is in sharp contrast with the labeled case where is was recently show in [1] that there exists labeled routing schemes with polylogarithmic storage factors and $O(1)$ average-stretch. Unlike the max-stretch case where with $\widetilde{O}(n^{1/k})$ storage[2] both labeled and name-independent models have max-stretch $\Omega(k)$, our results provides a clear separation on the average-stretch between labeled $\Theta(1)$ and name-independent routing schemes $\Theta(k)$.

1.2 Related work

Labeled routing on a trees is explored in [5, 14], achieving stretch 1 with $O(\log^2 n / \log \log n)$ bits for local tables and for headers, and this is tight [6]. Laing [9] 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 $2k-1$.

Independently from our work, Rajaraman and Laing [10] give a lower bound showing that max-stretch 3 name-independent schemes require $\Omega(\sqrt{n})$ bit of storage. Our results are stronger as they show that any max-stretch below 5 requires $\Omega(\sqrt{n})$ bit of storage. In addition, our results are parameterized for any integer $k \geq 1$, we note that the cases $k > 1$ are the ones that require the novel distributed Kolmogorov complexity argument. Finally our results apply to average-stretch as well.

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

1.3 Techniques

Our lower bound technique to prove Theorem 1 is new and is interesting in its own right. All previous lower bounds in the field of compact routing [13, 7, 8, 3, 14] 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, [13] and then [14] 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 [4] 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 $\Omega(n^{1+1/(3\lfloor k/2 \rfloor - 1)})$ [11]. Although it applies on both labeled and name-independent routing schemes, this technique cannot be used to prove Theorem 1, 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 [5, 14]. 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. [2]).

We present below the formal proof that makes an original use of Kolmogorov Complexity. We first define in Section 2 precisely the routing scheme model and its memory requirements. Then, in Section 3 we provide background on Kolmogorov Complexity, and we describe the counter-example in Section 4. Finally, the proof of the final theorem is given in Section 5.

2. ROUTING SCHEME MODEL AND MEMORY REQUIREMENTS

We use a general and standard model for routing schemes, originally introduced by Peleg and Upfal in [13]. A *routing scheme* is a function that associates with each graph G:

1. a *name* to each node u of G, denoted by $\text{name}(u)$;
2. a *port number* to each ordered pair (u, v) of adjacent nodes of G, denoted by $\text{port}(u, v)$, 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 1).

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.

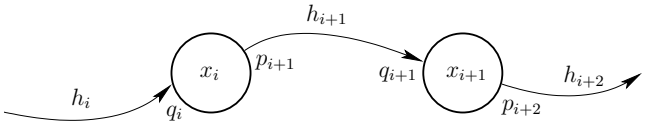

Figure 1: Model of a routing function: $R(x_i, h_i, q_i) = (h_{i+1}, p_{i+1})$**.**

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.

3. 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 [12] 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.

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 $L^* = \{L[i] \mid 1 \leq i \leq |L|\}$.

We first need the following technical lemma (see Section 7 for the proof):

Lemma 3. *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 $|A| \leq C(A) + 2\log n$; and*
2. *every sub-sequence B of L has $|B| \leq (2C(B) + 4\log n)/\log(C(B) + 2\log n)$.*

4. 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 3 (but with parameter $n-1$ instead of n). For short, we set $\ell = |L| = \lfloor (n-1)/2 \rfloor$. More precisely, for $y \in \{1, \ldots, \ell\}$, $\text{port}(r, y) = L[y]$, and all the other ports are fixed arbitrarily (but still keeping the conventions, in particular, all the ports of r are distinct and $\text{port}(y, r) = 1$ 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 $k \geq 1$ 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

[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.

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.

5. 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:

$$\begin{array}{llllll} h_0(1) & h_1(1) & h_2(1) & h_3(1) & & \\ h_0(2) & h_1(2) & & & & \\ h_0(3) & h_1(3) & h_2(3) & h_3(3) & h_4(3) & h_5(3) \\ \vdots & & & & & \\ q_0(1) & q_1(1) & q_2(1) & q_3(1) & & \\ q_0(2) & q_1(2) & & & & \\ q_0(3) & q_1(3) & q_2(3) & q_3(3) & q_4(3) & q_5(3) \\ \vdots & & & & & \end{array}$$

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

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 1. *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,

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

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.

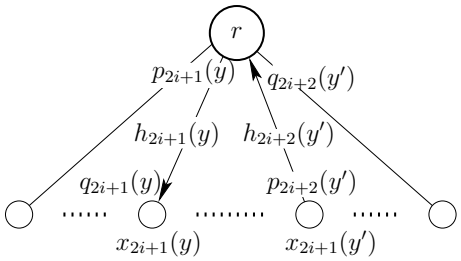

Figure 2: A step of the walks to y and y' in the star network.

Lemma 4. *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 5. *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 6. *There is a constant c such that $\forall i \geq 0$,*

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:

		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

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:

		y_1	y_3	y_4	y_5
Q_{2i+1}	=	2	3	1	2

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 2). 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 $\text{port}(r, y) = L[y]$ for all $y \in \{1, \ldots, \ell\}$. Let us denote by y_p the unique destination of $\{1, \ldots, \ell\}$ such that $\text{port}(r, y_p) = p$.

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 2). 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 $j = 1, \ldots, |H_{2i+1}|$

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

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. (1) is $t = |P^*_{2i+1}|$. So, given P_{2i+1}, H_{2i+1}, Eq. (1) 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 6. □

We have now all the material to prove Theorem 1. 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 6, 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 3 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 1, $i_{\max} < k$. 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 the following.

Theorem 1. *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 (the max-stretch) 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 2. *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 $w = C(X)$. □

Claim 3. *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 2, 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. □

Observe that, since $C(X \mid Y) \leq C(X)$, Claim 3 directly implies that $C(X,Y) \leq C(X) + C(Y) + 2\min\{\log C(X), \log C(Y)\} + c$.

Claim 4. *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 3, completing the proof of the claim. □

Plugging $Z = \varepsilon$ (the empty string) in Claim 4, 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, whenever $C(X \mid Y)$ is used in a formula, it stands for $C(X \mid Y, n, \ell, k, M)$.

Combining Claim 3 with Property 4 of Lemma 6, we obtain for every $i \geq 0$:

$$\begin{aligned} C(H_{2i+2}, P_{2i+2}) &\leq C(H_{2i+2}, P_{2i+2} \mid H_{2i+1}, P_{2i+1}) && (2)\\ +C(H_{2i+1}, P_{2i+1}) + 2\log C(H_{2i+2}, P_{2i+2} \mid H_{2i+1}, P_{2i+1}) && (3)\\ +O(1) &\leq |P^*_{2i+1}| \cdot (M+1) + C(H_{2i+1}, P_{2i+1}) && (4)\\ &\quad +2\log\left(|P^*_{2i+1}| \cdot (M+1)\right) + O(1) && (5)\end{aligned}$$

Since the stretch is $< 2k+1$, one can apply Lemma 3 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. (5), we obtain:

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

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

$$\begin{aligned} C(H_{2i}, P_{2i}) &\leq C(H_{2i-1}, P_{2i-1}) + && (6)\\ &\quad (M+1) \cdot (C(P_{2i-1}) + 2\log n) + && (7)\\ &\quad O(M + \log n) && (8)\end{aligned}$$

Combining Properties 2 and 3 of Lemma 6 with Claim 4, 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 3,

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

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 $i \geq 1$,

$$\begin{aligned} C(H_{2i+1}, P_{2i+1}) &\leq C(H_{2i}, P_{2i}) + C(P_{2i-1}) + && (9)\\ &\quad 2\log(n\log n) + O(M) && (10)\\ &\leq C(H_{2i}, P_{2i}) + C(P_{2i-1}) + O(M + \log n) && (11)\end{aligned}$$

Plugging Ineq. (8) in Ineq. (11), we obtain for every $i \geq 1$:

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

Combining Claim 3 with Property 3 (for $i = 0$), and Property 1 of Lemma 6 we have:

$$\begin{aligned} C(H_1, P_1) &\leq C(H_1, P_1 \mid H_0, Q_0) + C(H_0, Q_0) && (12)\\ &\quad +2\log C(H_0, Q_0) + O(1) && (13)\\ &\leq M + O(1) && (14)\end{aligned}$$

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:

$$\begin{aligned} C(P_{2i+1}) + 2\log n &\leq && (15)\\ (M+2)\sum_{j=1}^{i} (C(P_{2j-1}) + 2\log n + c(M + \log n)) && (16)\\ \leq (1+\varepsilon)(M+2)\sum_{j=1}^{i} (C(P_{2j-1}) + 2\log n) && (17)\end{aligned}$$

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. (17) rewrites in,

$$\forall\, i \geq 1, \quad C_{i+1} \leq (1+\varepsilon)(M+2)\sum_{j=1}^{i} C_j \qquad (18)$$

Claim 5. *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 5 holds for $i = 0$. Assume it holds for every $1 \leq j \leq i$. We have:

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

Therefore, by Ineq. (18), $C_{i+1} \leq (1+\varepsilon)(M+2)\sum_{j=1}^{i} C_j \leq (\alpha(M+2))^{i+1}$ completing the proof Claim 5. □

The stretch is $< 2k+1$, thus by Lemma 3 and Claim 5,

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

From the proof of Claim 5, 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 4, for M large enough, i.e., $M = \Omega(\log n)$, $\alpha > 1$, and $\varepsilon > 0$.

$$\begin{aligned}\forall i \geq 0, \quad |W_i| &\leq \sum_{j=0}^{i} |P^*_{2j+1}| && (19)\\ &< O(i) + \frac{\alpha^{i+2}}{1+\varepsilon}\cdot(M+2)^{i+1} && (20)\\ &< \alpha^{i+2}\cdot(M+2)^{i+1} && (21)\end{aligned}$$

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

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

from Lemma 5, $P_{2i_{\max}}$ 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 3 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 5, 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 1. 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 1.

6. AVERAGE-STRETCH FACTOR

The goal of this section is to prove the following:

Theorem 2. *For each integer $k \geq 1$ there is a weighted n-node star for which every name-independent routing scheme with less than $(n/(9k))^{1/k}$ bits of memory requirements has average-stretch at least $k/4 + 7/8$.*

For that we simply extended the previous lower bound. The main change is that the weight k in the star network is replaced by a larger weight K, say $K = n^2/2$ (the value can be optimized). The proof is in two parts: first we show that any single-source routing scheme R on the star (with source r) with memory $M = O(n^{1/k})$ has an average-stretch $\Omega(k)$. Then, we extended the result to any routing scheme on the star where the stretch is now averaging on the $n(n-1)$ pairs.

For a routing scheme R, let $\rho(x, y, R)$ denote the length of the route induced by R between x and y.

Let us consider any single-source routing scheme R on the star, with source r, and with memory requirements $M = M(n, k) = \Omega(\log n)$.

Lemma 7. *If $M < c_k \cdot n^{1/k}$, then $\sum_{y=1}^{\ell} \rho(r, y, R) > 2k\ell$, where $c_k = \frac{1}{2}(8k+4)^{-1/k}$.*

PROOF. To prove Lemma 7 we reuse the proof of Theorem 1, in particular Ineq. (19), that is: for $M = \Omega(\log n)$ with n large enough, $\alpha > 1$, and $\varepsilon > 0$,

$$\forall i \geq 0, \quad |W_i| \;<\; \alpha^{i+2}\cdot(M+2)^{i+1}$$

where $\alpha = 1+\varepsilon+1/(M+2)$. Choosing $\varepsilon = c/\log n$ for some suitable constant c, $\alpha^j \leq 2$ for every $j \geq 1$. Thus,

$$\forall i \geq 0, \quad |W_i| \;<\; 2\cdot(M+2)^{i+1}\,. \qquad (22)$$

This later equation holds under the condition that $P^*_{2i+1} \subseteq L^*$. Unfortunately, the first part of Lemma 5 does not hold, and so Ineq. (22) is wrong in general. However we have:

CLAIM 6. *If $\sum_{y=1}^{\ell} \rho(r, y, R) \leq n^2$, then $P^*_{2i+1} \subseteq L^*$.*

PROOF. Similarly to Lemma 5, assume that there is some $p \in P^*_{2i+1}$ such that $p \notin L^*$. Then, the port p leads to a node $w \notin \{1, \ldots, \ell\}$. Since P_{2i+1} concerns routing from r 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, $\rho(r, y_0, R) \geq 2K$. Thus $\sum_{y=1}^{\ell} \rho(r, y, R) > 2K = n^2$: a contradiction. □

If $\sum_{y=1}^{\ell} \rho(r,y,R) > n^2$, then Lemma 7 is true, since $2k\ell < n^2$ for large n. And, if $\sum_{y=1}^{\ell} \rho(r,y,R) < n^2$, then by Claim 6, $P^*_{2i+1} \subseteq L^*$. Therefore we can assume that Ineq. (22) holds.

Recall that $W_i \subseteq \{1,\ldots,\ell\}$ is the subset of destinations reached by R from r after a walk of length at most $2i+1$. Since for every $y \in \{1,\ldots,\ell\} \setminus W_i$, $\rho(r,y,R) \geq 2i+3$ and $d(r,y)=1$, we obtain:

$$\sum_{y=1}^{\ell} \rho(r,y,R) \;>\; \sum_{y\in\{1,\ldots,\ell\}\setminus W_i} \rho(r,y,R) \;\geq\; (\ell - |W_i|)\cdot(2i+3)$$

If $M < c_k \cdot n^{1/k}$, then, as $\ell = \lfloor (n-1)/2 \rfloor$, $2(M+2)^k < \ell/(2k+1)$ for n large enough. By Ineq (22), $|W_i| < 2(M+2)^{i+1} < \ell/(2k+1)$. For $i = k-1$, we obtain:

$$\sum_{y=1}^{\ell} \rho(r,y,R) \;>\; (2k+1)\cdot(\ell - \ell/(2k+1)) \;=\; 2k\ell\ .$$

completing the proof of Lemma 7.

We now consider any routing scheme R^* for the star. The average-stretch of R^* is:

$$\bar{s}(R^*) \;=\; \frac{1}{n(n-1)} \sum_{x\neq y} \frac{\rho(x,y,R)}{d(x,y)} \;=\; \frac{1}{n}\sum_{x} \bar{s}(R^*,x)$$

where $\bar{s}(R^*,x) = \frac{1}{n-1}\sum_{y\neq x} \rho(x,y,R)/d(x,y)$.

Given a node x, we simulate as follows a single-source routing scheme, named $\tilde{R}_x$, with source r and based on R^*. For every node $y \neq x$, $\tilde{R}_y$ works exactly as R^*_y. And, $\tilde{R}_r$ is composed of R^*_r, of R^*_x, and of the port number leading from r to x. Sending a message by $\tilde{R}$ from r to y is done as follows: if $y = x$, then the message is sent directly to the edge (r,x). If $y \neq x$, then function R^*_x is applied first (generating eventually suitable headers but without sending any message), and then R^*_r is applied and the message is then forwarded.

For each node x, the resulting routing scheme $\tilde{R}_x$ is a single-source routing scheme with source r on the star, and has memory requirements at most $2M^* + O(\log n)$, where M^* denotes the memory requirements of R^*.

By the simulation $\rho(x,y,R^*) = d(x,r) + \rho(r,y,\tilde{R}_x)$. Thus one can estimate the stretch factor of R^* by:

$$\begin{aligned}
&\forall x \in \{1,\ldots,\ell\}\,,\\
\bar{s}(R^*,x)\cdot(n-1) \;&=\; \sum_{y\neq x} \frac{\rho(x,y,R^*)}{d(x,y)}\\
&=\; \sum_{y\neq x} \frac{\rho(r,y,\tilde{R}_x) + d(x,r)}{d(x,y)}\\
&=\; \sum_{y=1}^{\ell} \frac{\rho(r,y,\tilde{R}_x)}{2} + \frac{\ell}{2} + \sum_{y=\ell+1}^{n-1} \frac{\rho(x,y,R^*)}{d(x,y)}\\
&\geq\; \frac{1}{2}\sum_{y=1}^{\ell} \rho(r,y,\tilde{R}_x) + \frac{\ell}{2} + (n-1-\ell)
\end{aligned}$$

Assume now that $M^* < (n/(9k))^{1/k}$. It follows that $2M^* + O(\log n) < c_k \cdot n^{1/k}$, and Lemma 7 can be applied: $\sum_{y=1}^{\ell} \rho(r,y,\tilde{R}_x) > 2k\ell$. Therefore, for every x:

$$\begin{aligned}
\bar{s}(R^*,x)\cdot(n-1) \;&>\; k\ell - \ell/2 + n - 1\\
&=\; n - 1 + \ell(k-1/2)\\
\Rightarrow\quad \bar{s}(R^*,x) \;&>\; k/2 + 3/4\\
\Rightarrow\quad \bar{s}(R^*) \;&=\; \frac{1}{n}\sum_{x} \bar{s}(R^*,x)\\
&=\; \frac{1}{n}\left(\sum_{x=1}^{\ell} \bar{s}(R^*,x) + \sum_{x=\ell+1}^{n-1} \bar{s}(R^*,x)\right)\\
&>\; \frac{1}{n}\left(\ell(k/2+3/4) + (n-1-\ell)\right)\\
&=\; k/4 + 7/8
\end{aligned}$$

completing the proof of Theorem 2.

7. PROOF OF LEMMA 3

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 $C(L \mid n) \geq \log\left(\ell!\binom{n}{\ell}\right)$ and $C(L^* \mid n) \geq \log\binom{n}{\ell}$. Note that $|L| = |L^*| = \ell$. Let us prove the first claim of Lemma 3. 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,

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

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 $|S_2| \leq n + O(1) \leq 2n$ 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, $C(L^* \setminus A \mid A) \leq \log\binom{n-a}{\ell-a} + O(1)$. Therefore, we have the upper bound for the Kolmogorov Complexity of L^* that is:

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

And, by the previous lower bound, it yields:

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

Set $m = C(A) + \log n + O(1)$ so that Ineq. (23) rewrites:

$$\binom{n}{\ell} \;\leq\; \binom{n-a}{\ell-a}\cdot 2^m \quad\Leftrightarrow\quad \frac{n!}{\ell!} \;\leq\; \frac{(n-a)!}{(\ell-a)!}\cdot 2^m \qquad (23)$$

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:

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

Plugging in Ineq. (23) the later lower and upper bound, we obtain that:

$$\frac{n^n}{\ell^\ell} \;\leq\; \frac{(n-a)^{n-a}}{(\ell-a)^{\ell-a}} \cdot 2^m \sqrt{2\pi(n-a)} \tag{24}$$

Let us show that:

Claim 7. *For all integers a, ℓ, n such that $0 \leq a \leq \ell$ and $0 < \ell \leq n$,*

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

Proof. This inequality is equivalent to

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

where $f(x) = (1-a/x)^x$ with $x \geq a$ and $x \neq 0$. It remains to show that f is decreasing for every $x \geq a$. 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 $f'(x) \leq 0$ for every $x \geq a$, and so f decreases as claimed. □

Assume now that $2^m\sqrt{2\pi n} < n^a/\ell^a$. By Claim 7, 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. (24). Therefore, we have $2^m\sqrt{2\pi n} \geq (n/\ell)^a$, that is

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

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

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

completing the proof of the first claim.

We now prove the second claim of Lemma 3. 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 $S = S_1S_2S_3S_4$ 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:

$$\log\left(\ell!\binom{n}{\ell}\right) \;\leq\; \log\left((\ell-b)!\binom{n-b}{\ell-b}\binom{\ell}{b}\right) \tag{25}$$
$$+\; C(B) + \log(n\log n) + O(1)\;. \tag{26}$$

Setting $t = C(B) + \log(n\log n) + O(1)$, and using the fact that $(\ell-b)!\binom{\ell}{b} = \ell!/b!$, Ineq. (25) 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. (23),

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

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

$$b + \log(b!) \;\leq\; t + \log\sqrt{2\pi n}$$
$$\Rightarrow\; b(1 + \log(b/e)) \leq t + \log\sqrt{2\pi n}$$
$$\Rightarrow\; b\log(2b/e) \;\leq\; C(B) + 2\log n$$

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

$$\frac{b\log(2b/e)}{\log(b\log(2b/e))} \;\leq\; \frac{C}{\log C}\;. \tag{27}$$

Note that $\log x \leq 4x/e$ for every $x \geq 1$. So $b\log(2b/e) \leq b \cdot 4b/e^2 = (2b/e)^2$. Hence, Ineq. (27) implies that

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

proving the second claim, and completing the proof of Lemma 3.

8. REFERENCES

[1] Ittai Abraham, Yair Bartal, and Ofer Neiman. Advances in metric embedding theory. In 38^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 271–286. ACM Press, May 2006.

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

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Kofi Ambrose Laing and Rajmohan Rajaraman. Brief announcement: A space lower bound for name-independent compact routing in trees. In 17^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, page 216. ACM Press, July 2005.

[11] 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.

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

[13] 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.

[14] 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.