Laboratoire Bordelais de Recherche en Informatique
Université Bordeaux I, 351, cours de la Libération,
33405 Talence CEDEX, France

Research Report `RR-1181-97`

Space-Efficiency for Routing Schemes of Stretch Factor Three

by Cyril Gavoille and Marc Gengler

October, 1997

Space-Efficiency for Routing Schemes of Stretch Factor Three*

Cyril Gavoille† Marc Gengler‡

Abstract

We deal with *routing algorithms* on arbitrary n-node networks. A *routing algorithm* is a deterministic distributed algorithm which routes messages from any source to any destination. It includes not only the classical routing tables, but also the routing algorithm that generates paths with loops. Our goal is to design routing algorithms which minimize, for each router of the network, the amount of routing information that needs to be stored by the router in order to implement its own local routing algorithm. So as to simplify the implementation of a routing algorithm, names of the routers can be chosen in advance. We take also into account the efficiency of the routing, i.e., the length of the routing paths. The *stretch factor* is the maximum ratio, taken over all source-destination pairs, between the length of the paths computed by the routing algorithm and the distance between the source and the destination. We show that there exists an n-node network on which every routing algorithm of stretch factor $s < 3$ requires at least a total of $\Omega(n^2)$ bits of routing information, whereas for stretch factor $s = 3$ the best known upper bound is $O(n^{3/2} \log^2 n)$ bits in total [1]. We show a similar gap for the space complexity of routing schemes on the subclass of networks of diameter 2.

Keywords: compact routing in distributed networks, near-shortest path routing, routing tables.

---

*A preliminary version has been partially published in the proceedings of SIROCCO '97.

†LaBRI, Université Bordeaux I, 351, cours de la Libération, 33405 Talence Cedex, France. e-mail: *gavoille@labri.u-bordeaux.fr*

‡LIP, École Normale Supérieure de Lyon, 69364 Lyon, Cedex 07, France. e-mail: *Marc.Gengler@ens-lyon.fr*

1 Introduction and Model

Routing messages in networks is an important task. For traffic regulation or for minimization of the delivery time of messages, we need routing paths that are the shortest possible or that are almost the shortest ones. A routing path is said to be "almost the shortest" if the ratio between the length of the path and the distance between the source and the destination is bounded by a constant, called the stretch factor of the path. This definition is naturally extended to a network G and a routing function R for G is of stretch factor s if the stretch factor of no routing path exceeds s. More formally, the stretch factor of a routing R on a network G is defined by:

$$s(R, G) = \max_{x \neq y} \frac{d_R(x, y)}{d_G(x, y)}$$

where $d_R(x, y)$ is the routing path length between x and y, and $d_G(x, y)$ the distance in G.

For arbitrary networks *routing tables* are commonly used. Each router has a table with n entries. Messages for a node labeled i is forwarded to the port number which is read at the i-th entry of the table. Such a technique is *universal*, usable for all kinds of topologies, and can generate minimal routing path lengths. On the other hand, each router needs to store locally $O(n \log n)$ bits. This leads to $O(n^2 \log n)$ bits for the whole network, which is not a reasonable amount of memory, especially for growing networks. However, one can hope to do better because, to a certain extent, topologies are not always arbitrary.

Routing in a ring, for example, requires a simple algorithm for each router. A circular labeling of nodes allows to code a "distance information" for each destination – see [3] for notion of *sense of direction* in networks. A router x forwards a message for destination y by sending it to the left or to the right output port, simply by comparing y and $x + \lceil n/2 \rceil \bmod n$, which is the node at maximum distance from x. Such a routing algorithm allows a compact implementation with low amount of hardware, using $O(\log n)$ bits, mainly because the names of routers can be chosen in advance. Note also that the number of bits locally required strongly depends on the memory complexity measure. Clearly, if the labeling of the nodes cannot be chosen in advance, the routers in a ring could require $\Theta(n)$ bits. In [2], many models of memory complexity are described with results for lower and upper bounds for the average case. In this paper, we choose one of the strongest models, which takes the names of the nodes from the set $\{1, \ldots, n\}$ and allows output port relabeling. A stronger model is obtained when the names of routers can have arbitrary size. However, with names of unlimited size the memory complexity may drop to $O(\log n)$ bits per router, as the name of a router could contain all the routing information of the network and it would be sufficient for any router to store only an ID on $O(\log n)$ bits. The routing algorithm would be reduced in reading the output port directly from the header containing name of destination with the entire routing table.

It is straightforward to show that $\Omega(n)$ bits are necessary to store any neighbor-optimal routing table if no node-labeling is allowed. Think, for instance, of a router which is connected to $n/2$ nodes chosen randomly and which must be able to determine its neighborhood. However, if name and output port relabeling is allowed, $\Omega(n)$ bits are not always required to optimally route to neighbors. Consider, as an example, the complete graph K_n and assume that, for each node x, the output port numbered i, for every $i \in \{1, \ldots, n-1\}$, reaches node y which has the i-th lowest label among the $n-1$ neighbors of x. With this labeling, a message from x to one of its neighbors y uses output port y, if $y < x$, and $y-1$, otherwise. Thus, it is sufficient for any router x to know its name, yielding a local storage of only $O(\log n)$ bits. Note, also, that $n + O(\log n)$ bits per node are sufficient to optimally route to the neighbors on any network. Each router x stores an n-bits vector which codes, for any node y, whether y is a neighbor of x by setting the x-th bit appropriately. The number of the output port which reaches y is chosen as the number of neighbors of x whose label is less or equal to y. $O(\log n)$ bits are added to determine whether $x = y$.

If many shortest paths are available in the network, a smart choice of shortest paths may contribute to compress the routing information. For instance, in the complete bipartite graph, $K_{2,n-2}$, $\Theta(n)$ bits can be required for routers of the $n-2$ partition, if routing paths are randomly selected among the two shortest paths. Clearly, a better distribution of the routing shortest paths allows to store some shortest path routing function on $K_{2,n-2}$ in $O(\log n)$ bits per router.

Our result is proved for a model of routing function described in [5]. A routing algorithm, or a *routing function* R, is a pair (P, H) consisting of a *port function* P and a *header function* H. For any two distinct nodes u and v, R produces a path $u = u_0, u_1, \ldots, u_k = v$ of nodes, a sequence $h_0, h_1, \ldots, h_k$ of headers and a sequence $p_0, p_1, \ldots, p_k$ of output port numbers. A message with header h_i and arriving at node u_i through input port p_i is forwarded to the output port $P(u_i, p_i, h_i) = p_{i+1}$. Its new header will be $H(u_i, p_i, h_i) = h_{i+1}$. Thus, we require that for all i, $i \in \{1, \ldots, k-1\}$, $H(u_i, p_i, h_i) = h_{i+1}$, $P(u_i, p_i, h_i) = p_{i+1}$ and that p_{i+1} corresponds to the output port number of the link (u_i, u_{i+1}). On each router, the input and output ports numbered 0 are associated with the special links between the router and its host. This allows us to complete the description by imposing that $p_0 = 0$, $P(u_k, p_k, h_k) = 0$ as well as $h_0 = v$, fixing the initial header provided to the routing function. We are interested in implementing a distributed version of the routing functions. So, routing functions are expressed in this paper as collections of local routing functions $R_x = (P_x, H_x)$, indexed by the names x of the nodes. Finally, a *routing scheme* is a function that returns a routing function R for every n-node network.

Our contribution is a lower bound of $\Omega(n)$ bits of local storage needed to implement any local routing function of any stretch factor $s < 3$ in a worst-case n-node network. Furthermore, we show that this situation occurs on $\Theta(n)$ routers of the network. The lower bound is extended to networks of diameter 2. All these results are established in a context where node and port relabeling is allowed, yielding bounds that are valid for whatever choice of the node or port names.

2 Previous Work and Examples

Let us present the best known upper bound for stretch factor 3 and recall the routing scheme used by Awerbuch, Bar-Noy, Linial, and Peleg in [1, page 321]:

Theorem 1 (Awerbuch, Bar-Noy, Linial, Peleg) *Every n-node graph has a routing algorithm of stretch factor at most 3 with a total memory requirement $O(n^{3/2}\log^2 n)$ bits.*

The case considered here is $s = 3$. The result is established as follows. Let G be an n-node graph. The nodes are assumed to be labeled by unique integers taken from $\{1,\dots,n\}$, and output ports of each node x are taken from $\{1,\dots,\deg(x)\}$, where $\deg(x)$ is the number of neighbors of x. The idea is to choose in G a set of $O(\sqrt{n}\log n)$ nodes, called *pivots*, such that each node of G has at least one pivot in a *neighborhood* of $O(\sqrt{n})$ nodes of G. A neighborhood of x is a set of closest nodes of x. Awerbuch *et al.* showed that such a set of pivots can always be chosen. At each pivot p is rooted a shortest path spanning tree T_p. The scheme is to route directly with a full shortest path routing table if the destination is in the neighborhood of the sender, otherwise to route to the closest pivot, which will route then along a shortest path to the final destination. By the triangle inequality, the length of the routing path is at most 3 times longer than the distance. To route along each tree T_p we can use an *interval routing scheme* that consists in labeling the nodes with a depth first scheme, and in storing a table of intervals. Each interval groups the destinations that uses the same outport port number. Each tree T_p contributes for a total of $O(n\log n)$ bits: one interval for each edge of T_p is used, and $O(\log n)$ bits suffices to code the boundaries of the interval and the output port number assigned to the interval. Each pivot needs to give an own labeling of the nodes for its interval routing. So, each pivot p stores a translation table which converts each destination y into the labels used for T_p. This costs an overhead of $O(n\log n)$ bits for each pivot. In the same time each other node x stores $O(\sqrt{n}\log n)$ interval tables, each one corresponding to an interval routing of a different pivot. Moreover, x stores a shortest path routing table for its neighborhood, on $O(\sqrt{n}\log n)$ bits, and also the output port number of its closest pivot p_x. Indeed, x stores a table of $O(\sqrt{n})$ entries of $O(\log n)$ bits which gives the names of the nodes in the neighborhood and their output port numbers. This gives a storage of $O(n^{3/2}\log^2 n)$ bits for the entire network. The route from x to y is a shortest path if y is in the neighborhood of x, as x knows from its table whether y is in its neighborhood. Otherwise, x routes towards its closest pivot, p_x. During this step the header contains two parts: y and p_x. At p_x, y is converted into the local name of the spanning tree T_{p_x}, and the header contains now p_x and the translated name of y, say y'. The routing scheme routes to the label y' along a shortest path using the p_x-th interval table of each intermediate node.

We will see that the problem of routing with stretch factor $s < 3$ requires $\Omega(n^2)$ bits, significantly much more bits than for $s = 3$. The $\Omega(n^2)$ lower bound recalls the minimum number of bits to code a n-node network. Indeed, the number of (unlabeled) topologies with n nodes is $2^{n^2/2+o(n^2)}$ [12], showing that some topology needs at least $\Theta(n^2)$ bits to be described. Note that $n^2/2$ bits are sufficient by the use of the $n \times n$ adjacency matrix of the graph. However, this counting argument neither gives any efficient way to distribute the routing information among the n routers, nor can it be applied to obtain a lower bound. The next examples will show that the number of n-node networks does not give in general any indication for memory complexity of the routing function, especially if shortest paths are not required.

For the sake of simplicity, examples are given in the weaker model of routing tables. $R(x, y)$ returns the number output port on router x for a message with destination y without any possibility to modify the header. Note that labeling of nodes and ports is allowed to start with 0 for this example.

Example 1: *xor*-routing.

$R(x, y)$ ends the protocol if $x = y$. Otherwise, it sends the message to the port number x, where x is the position of the first non null bit in x *xor* y. In the 3-cube depicted on the left of Figure 1, the routing path from 101 to 010 uses successively ports 1 to go to node 100, port 2 to go to node 110 and finally port 3 to reach its destination. This is the routing algorithm used on hypercube-based architectures. It is easy to check that *xor*-routing applies to both topologies of Figure 1, the 3-cube and the twisted 3-cube. On the twisted 3-cube, the routing path from 101 to 010 uses port 1 to go to node 100, port 2 to go to node 111, port 1 to go to node 110 and port 3 to go to node 010, yielding a path that is not the shortest possible.

xor-routing does not always generate shortest paths on the twisted 3-cube, but Kranakis, Krizanc, and Urrutia showed in [8] that it is possible to find a slightly different routing algorithm providing shortest path routing on both topologies.

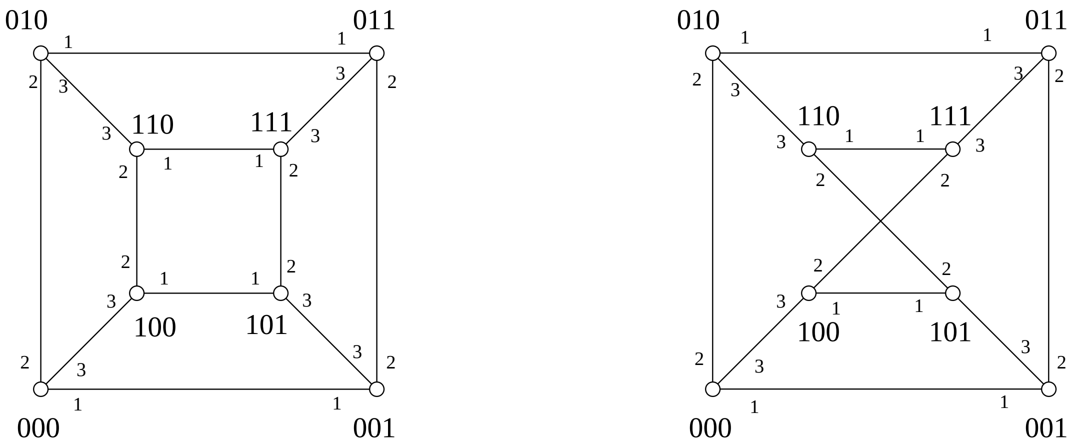


Figure 1: The *xor*-routing.

Example 2: *exp*-routing.

The *exp*-routing algorithm applies to topologies as depicted on Figure 2 or to the ones of Table 1. $R(x, y)$ ends the protocol if $x = y$. Otherwise, it computes $\exp(x)$ (radix 3) and extracts the y-th digit, which gives the output port number. The integer part of $\exp(x)$ is not taken into account and the digits of the fractional part are numbered starting with 1. For instance, $R(4, 2) = 2$ because $e^4 = 2000.1\mathbf{2}1011\ldots_3$ and the second digit is "2". Table 1 gives the first values of the routing table for the *exp*-routing. Note that the stretch factor of the *exp*-routing for the examples of Figure 2 is 2, whereas it is 4 and 5 (path from the node 3 to the node 1) for the ones of Table 1.

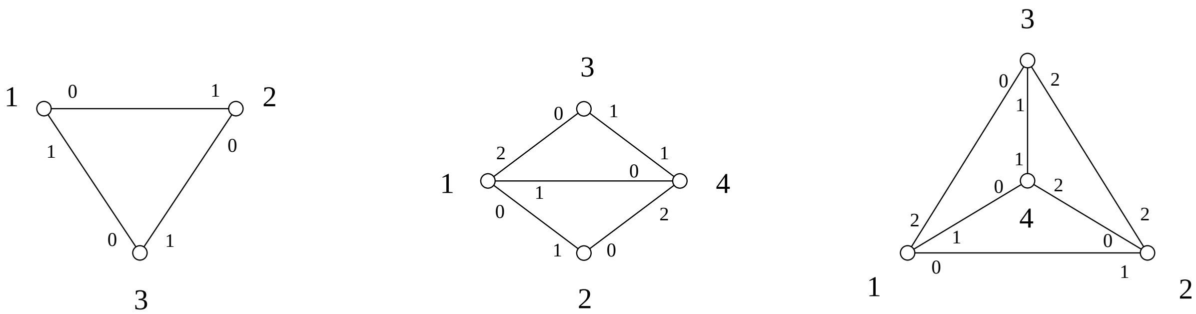
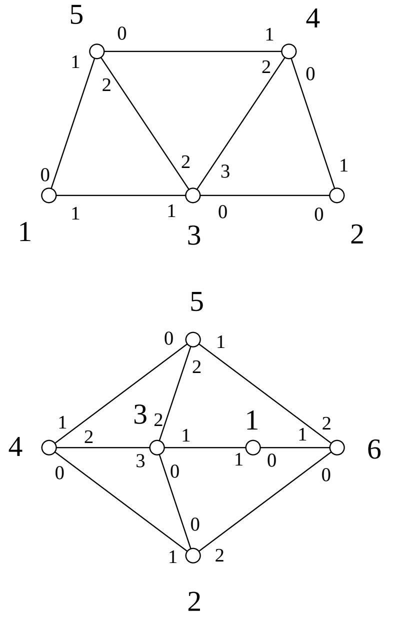


Figure 2: The *exp*-routing.

This example suggests that the "routing isomorphism" may happen on very different kinds of topologies. The reader must be convinced that many other *exp*-routing algorithms, based on computable irrational numbers, may be given. One could design, for instance, the π-routing algorithm, where $R(x, y)$ ends the protocol if $x = y$, and returns the xy-th digit of π, otherwise. All these routing schemes have the property that they can be computed based only on the



			1234567...
e^{1}	=	2.	2011011...
e^{2}	=	21.	1011111...
e^{3}	=	202.	0020221...
e^{4}	=	2000.	1210110...
e^{5}	=	12111.	1020110...
e^{6}	=	112221.	1021201...

Table 1: Table for the *exp*-routing (radix 3).

knowledge of the name of the router and a small $O(1)$ size algorithm. Moreover, they are uniform for all the routers. This shows that non-structured and various topologies can support very simple routing algorithm.

Therefore, the number of routing functions generated by a routing scheme is smaller than the number of topologies, which may hurt intuition. Peleg and Upfal [10] described an example of non-isomorphic graphs supporting a same shortest path routing table. More fundamentally, Peleg and Upfal proved in [10] that $2^{\Omega(n^{1+1/(2s+4)})}$ routing functions are required to route with stretch factor at most s on the entire class of n-node networks, implying a lower bound of $\Omega(n^{1+1/(2s+4)})$ bits for the total memory requirements.

In [4], Fraigniaud and Gavoille improved the Peleg-Upfal's lower bound by proving the existence of $2^{\Omega(n^2)}$ non-isomorphic routing functions of stretch factor $s < 2$, including the interesting case of shortest path routing schemes ($s = 1$). Recently, Krizanc and Kranakis [7] reformulated the proof of [4] in term of Kolmogorov Complexity and showed also a lower bound of $\Omega(n^2)$ bits for $s < 2$. For stretch factor 1, Gavoille and Pérennès showed in [5] an optimal lower bound of $\Theta(n^2 \log n)$ bits, while Buhrman, Hoepman, and Vitányi showed in [2] that $O(n^2)$ bits are sufficient for shortest path routing on almost[1] all networks.

[1]They showed that a fraction of $1 - n^{-3}$ of all n-node labeled graphs supports routing schemes with $O(n^2)$ bits in total.

For stretch factor $s \geq 2$, no tight lower bound is known. For stretch factor $s = 3$, Peleg-Upfal's lower bound gives $\Omega(n^{1.1})$ bits, whereas $O(n^{3/2} \log^2 n)$ bits suffices [1]. In the next section, we will show that $\Omega(n^2)$ are necessary for any stretch factor $s < 3$.

3 Lower Bound of the Memory Requirement

We will prove a lower bound of the total number of bits required to describe a routing function of stretch factor less than 3 on n-node networks. The model considered for the definition of the routing function is slightly stronger than the one of the previous examples. In particular, headers can be modified and routing paths may include loops. Let us recall the model.

- The names of the nodes are unique integers taken in the set $\{1, \ldots, n\}$. For every node x, the numbers of the ports are unique integers taken in the set $\{1, \ldots, \deg(x)\}$, where $\deg(x)$ is the number of neighbors of x.
- Headers are of unlimited size. However, the first routing decision, in the router of the source, must be taken knowing only the name of the destination. This corresponds to the fact that the host machine gives only the name of the destination to its router and is modeled by $h_0 = v$ and $p_0 = 0$ in the definition in Section 1. The router of the sender takes its routing decision based on this minimal information and cannot take advantage of additional information, like a routing algorithm, provided by the host together with the message.
- Every router x can modify the header of the message before forwarding it.

The topology of the network is represented by an undirected, unlabeled, connected graph. Given a graph $G = (V, E)$ and a routing algorithm $R = \{R_x \mid x \in V\}$ on G, we define the *memory requirement of node x* as the length of the smallest program that computes the local routing function R_x. The length of the program is expressed in bits for a fixed programming language. The memory requirement is a measure of the routing information needed by the router x. It can be formally defined in terms of Kolmogorov Complexity [9]. The *local memory requirement of R* is the largest memory requirement of a node, taken over all the nodes of G. Finally, the *total memory requirement of R* is the sum of the memory requirements of all the nodes of G.

Note that the name of the router and the names of its output ports are not explicitly taken into account for the memory requirement. If nodes are not allowed to send messages to themselves via the routers (the host machine is assumed to have enough resources to manage such queries), simple shortest path routing functions on graphs like $K_{1,n-1}$ can use $O(1)$ bits[2] only, for any n. From a technical point of view, features like node/port relabeling and the choice of the shortest paths make harder lower bound proofs.

The basic idea is to show that some graphs of girth 4 cannot optimally route messages to their neighbors if all the routers store only $o(n)$ bits[3] of information. It follows that, certain routers, call one of them x, that have to send a message to some of their neighbors, say y, will forward the message in a wrong direction to some node z, $y \neq z$. By construction of the network the distance between y and z is at least 2. Either, the message will return from z to x before reaching its destination y in at least 3 steps, or the message will route from z to y along some other path of length at least 2. In all cases, the stretch factor is at least 3. So, some routers need at least $\Omega(n)$ bits to route with stretch factor strictly less than 3. Actually, we show that this situation may happen independently on $\Theta(n)$ routers, yielding a total of $\Omega(n^2)$ bits on the whole network.

We need the following combinatorial result to prove the main theorem.

Lemma 1 *For every integer $q \geq 1$ we have:*

$$\binom{q}{4q/5} \Big/ \binom{2q/5}{q/5} \;>\; \gamma \left(\frac{5}{4}\right)^q, \quad \text{with } \gamma \approx 0.65.$$

Proof. Using Formula (9.91) of [6] on page 481, derived from Striling's formula, we have for every $n \geq 1$,

$$\left(\frac{n}{e}\right)^n \sqrt{2\pi n} \;<\; n! \;<\; \beta \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$$

where $\beta = e^{1/12-1/360+1/1260} \approx 1.08$. It follows that for each $q \geq 1$, and for every α, $0 < \alpha < 1$,

$$\frac{H(\alpha)^n}{\sqrt{2\pi\alpha(1-\alpha)n}} \;<\; \binom{n}{\alpha n} \;<\; \beta\, \frac{H(\alpha)^n}{\sqrt{2\pi\alpha(1-\alpha)n}} \tag{1}$$

where H is the function defined by $H(\alpha) = \alpha^{-\alpha}(1-\alpha)^{-(1-\alpha)}$. We can upper-bound the term

$$\binom{2(1-\alpha)q}{(1-\alpha)q} \;<\; \beta\, \frac{H(1/2)^{2(1-\alpha)q}}{\sqrt{\pi(1-\alpha)q}}.$$

[2] For example, by labeling the node of degree $n-1$ with the number n, and its ports with the labels of its neighbors.

[3] We use the notation $f(n) = o(g(n))$ to express that functions f and g satisfy $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

Because $H(1/2) = 2$, we can therefore lower-bound the ratio by:

$$\binom{q}{\alpha q} \Big/ \binom{2(1-\alpha)q}{(1-\alpha)q} \;>\; \frac{1}{\beta\sqrt{2}} \left(\frac{H(\alpha)}{2^{2(1-\alpha)}}\right)^q .$$

By choosing $\alpha = 4/5$, we obtain $H(4/5)/2^{2/5} = 5/4$. It follows that

$$\binom{q}{4q/5} \Big/ \binom{2q/5}{q/5} \;>\; \frac{1}{\beta\sqrt{2}} \left(\frac{5}{4}\right)^q \;>\; 0.65 \left(\frac{5}{4}\right)^q .$$

□

In the next theorem we will use Lemma 1 by observing that the logarithm of the ratio of the binomial coefficients is linear in q. Theorem 3, which is a refinement of Theorem 2, requires a technical lemma, Lemma 2. As a first step we present Theorem 2.

Theorem 2 *For every sufficiently large integer n, there exists an n-node graph on which every routing algorithm of stretch factor $s < 3$ has a total memory requirement of $n^2/25$ bits at least.*

Proof.

Let $p = \lfloor n/4 \rfloor$ and $q = n - 2p \sim$[4] $n/2$. Let c be an integer such that $q/2 < c < q$. Let $\mathcal{M}$ be the set of $p \times q$ boolean matrices having exactly c 1-entries for each row. The cardinality of $\mathcal{M}$ is $|\mathcal{M}| = \binom{q}{c}^p$. For each matrix $M = (m_{i,j})$ of $\mathcal{M}$ we build a graph $G = (V, E)$. $V = W \cup A \cup B$, where $W = \{v_1, \ldots, v_q\}$, $A = \{a_1, \ldots, a_p\}$, and $B = \{b_1, \ldots, b_p\}$. For every $i \in \{1, \ldots, p\}$, a_i is adjacent to b_i, and for every $j \in \{1, \ldots, q\}$, v_j is adjacent to a_i if $m_{i,j} = 1$, or to b_i if $m_{i,j} = 0$. In other words, the i-th row of M, for any i, is a bit-vector of length q that indicates which nodes of W are neighbors of a_i. G has $2p + q = n$ nodes, and $p(q+1) \sim n^2/8$ edges. By construction the distance between any pair of nodes in W is at least 2, as any path connecting these nodes uses at least one intermediate node in $A \cup B$. See Figure 3 for a component $\{a_i, b_i\}$.

Using Figure 3, we can illustrate the principle of our construction as follows. Assume that a_i sends a message to one of its neighbors in W, say v_j. If a_i forwards to message to some other node, either to b_j or to some $v_k \in W$ with $k \neq j$, the resulting path will be of length at least 3, which is not admissible with $s < 3$. This reasoning applies independently to all nodes in A. Thus, any routing function R with $s < 3$ must route messages from any node $a_i \in A$ to one

[4]We use the notation $f(n) \sim g(n)$ to express that functions f and g satisfy $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$, or equivalently $f(n) = g(n) + o(g(n))$.

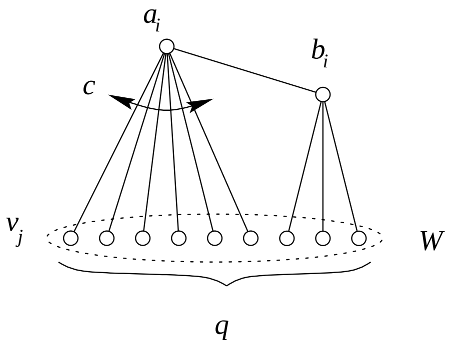


Figure 3: A router a_i with its c connections in the set W.

of the neighbors of a_i in W as if the stretch factor was 1, i.e., using shortest path routing.

Let $M \in \mathcal{M}$, and let G be the graph built from M by the previous construction. Let $R = \{R_x \mid x \in V\}$ be any routing algorithm of stretch factor $s < 3$ on G. Let $\mathrm{MR}(x)$ be the local memory requirement of x. The nodes and the ports of the nodes of G are assumed to be optimally labeled.

We assume that the following objects are given and show that their knowledge is enough to compute exactly M.

- the integers n, c, p, and q;
- the sequence $(v_1, \ldots, v_q)$ of the labels of nodes of W;
- the set $F = \{R_x \mid x \in A\}$ of routing functions;
- a permutation π of $\{1, \ldots, p\}$;
- a sequence $T = (t_1, \ldots, t_p)$ of integers.

Without loss of generality, we may assume that the routing functions of the set F are ordered lexicographically, i.e., $F = \{F_1, \ldots, F_p\}$. The permutation π is defined by $\pi(i) = j$ if and only if $F_j = R_{a_i}$, for every $i \in \{1, \ldots, p\}$. The role played by the sequence T will be given later. Let us now describe how to rebuild M.

(1) For each $v \in W$ and each $x \in A$, store in a $p \times q$ integer matrix $M' = (m'_{i,j})$ the integer values returned by $R_x = (P_x, H_x)$ at the first routing decision, that is when the input port is 0. More precisely, for every $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, q\}$, we set $m'_{i,j} = FP_{\pi(i)}(0, v_j)$, where FP_i is the port function of the routing function $F_i \in F$.

We observe that, if the stretch factor were $s = 1$, each row of M' would use all the $c + 1$ output ports of node a_i and that all the ports, except one, would occur exactly once. See Figure 3. From this, we could immediately rebuild M as the *unique-occurrences* correspond to the neighbors of a_i in W. However, with a stretch factor $s > 1$, messages from a node a_i to some node in W that is not a neighbor of a_i may initially be sent to another node than b_i, thereby perturbing the unique-occurrences. We now show how to overcome this problem.

From this step we apply independently the same procedure over all the rows of M'. Let us fix L as being the i-th row of M'.

(2) From L, compute U, the vector of *unique-occurrences* of L. That is, for every $j \in \{1, \dots, q\}$, the j-th bit of U is set if and only if the j-th entry of L appears once in L.

(3) Let z be the number of 0-entries of U, that is the number of entries of L that are not unique. Enumerate all the subsets of $\{1, \dots, z\}$ of size $c-(q-z)$. Without loss of generality, the enumeration is assumed to be ordered lexicographically. We use now the sequence T. Take the t_i-th subset enumerated, which is for instance B. B represents the set of positions of the 0-entries of U to flip.

(4) For every $j \in B$, flip the j-th 0-entry of U. The binary vector obtained is denoted by X.

We claim that X is exactly the i-th row of M. So, doing similarly over all the rows, we can rebuild M. Step (1) stores in M' the answers of the routers of A for the nodes of W. Call L the i-th row of M'.

Claim 1. The number of unique-occurrences of L is at least $2c - q$.

L has at least c distinct entries, otherwise the routing function R would not be neighbor-optimal, and thus of stretch factor $s \geq 3$. Therefore, at most $q - c$ entries can disturb in L the c distinct entries originally in the i-th row of M. In all cases, there are at least $c - (q - c) = 2c - q$ entries of L with unique-occurrences. Note that the number of unique-occurrences is a nonnegative integer since by assumption $c > q/2$.

Step (3) tries to determine the entries of L that are not "correct", that is the answers of the router a_i that do not provide a shortest path. As already said, if a_i would route with only shortest paths, U would characterize the neighbors of a_i in W and correspond exactly to the i-th row of M. Unfortunately, the stretch factor of R could be $1 < s < 3$, allowing routing paths from a_i to some node of W not neighbor of a_i through some neighbor of a_i in W. After Step (2), the bits set in U are correct, i.e., correspond to bits set in the i-th row of M. The number of correct bits is $q - z$, and thus the number of bits that remains to find is $c-(q-z)$. Since the bits to restore are among the 0-entries of U, it is sufficient to describe the right set of positions to flip to rebuild the i-th row of M entirely. The number of ways to do it is $\binom{z}{c-(q-z)}$. B codes the positions set to flip in the 0-entries of U. Let us now compute the total number of bits used for this construction.

Claim 2. Every finite sequence $S = (s_1, \ldots, s_k)$ of k integers such that $1 \le s_i \le m$, can be described, knowing k and m, by at most $k \log m + O(1)$ bits[5].

Indeed, it is sufficient to enumerate all such sequences – there are at most m^k – and to give the index of S in this enumeration. We need $\lceil k \log m \rceil$ bits for the index of S, and a constant number of bits to describe the enumeration algorithm.

Claim 3. $z \le 2(q-c)$.

The number of unique-occurrences, that is 1-entries in U, is $q-z$. By Claim 1, $q - z \ge 2c - q$, which implies $z \le 2(q-c)$.

Claim 4. $\binom{z}{c-(q-z)} \le \binom{2(q-c)}{q-c}$.

$\binom{z}{c-(q-z)}$ is maximum for z maximum if $c-(q-z) \le z/2$, for every z. By Claim 3, $z \le 2(q-c)$, thus $c-(q-z) \le z/2$. Using again Claim 3, we obtain $\binom{z}{c-(q-z)} \le \binom{2(q-c)}{q-c}$.

The integers n, c, p and q can be described with $O(\log n)$ bits. By Claim 1, the sequence $(v_1, \ldots, v_q)$ can be stored on $q \log n + O(1)$ bits. The permutation π can be stored on $p \log p + O(1)$ bits. Finally, T can be described by $p \log \binom{2(q-c)}{q-c} + O(1)$ bits using Claim 2 and Claim 4. Note that $\binom{2(q-c)}{q-c}$, that is the maximum of the sequence T, is entirely determined by the knowledge of q and c.

Let K be the number of bits to store the set F of routing functions. Since for every $M \in \mathcal{M}$ we can rebuild M, it follows that:

$$K + p \log \binom{2(q-c)}{q-c} + q \log n + p \log p + O(\log n) \ \ge\ \log |\mathcal{M}|.$$

Because $p, q \le n$, this can be rewritten in:

$$K \ \ge\ \log \left(|\mathcal{M}| \Big/ \binom{2(q-c)}{q-c}^p \right) - O(n \log n).$$

On the other hand, we have $K \le \sum_{x \in A} \mathrm{MR}(x) + O(\log \mathrm{MR}(x))$. K can be described by the sequence of routing functions R_x, each one stored in a self-delimiting way on $\mathrm{MR}(x) + O(\log \mathrm{MR}(x))$ bits. Note that $\mathrm{MR}(x)$ is bounded by a polynomial function in n, thus $\sum_{x \in A} \log \mathrm{MR}(x) = O(p \log n)$. Therefore, we get:

$$\sum_{x \in A} \mathrm{MR}(x) \ >\ \log \left(|\mathcal{M}| \Big/ \binom{2(q-c)}{q-c}^p \right) - O(n \log n). \tag{2}$$

By choosing $c = 4q/5$, we obtain $|\mathcal{M}| = \binom{q}{c}^p = \binom{q}{4q/5}^p$. Hence:

$$\sum_{x \in A} \mathrm{MR}(x) \ >\ \log \left(\binom{q}{4q/5} \Big/ \binom{2q/5}{q/5} \right)^p - O(n \log n).$$

[5]All logarithms are given radix 2.

By Lemma 1, we have:

$$
\begin{aligned}
\sum_{x\in A} \mathrm{MR}(x) &> pq\,\log\left(\frac{5}{4}\right) - O(n\log n)\\
&> n^2\left(\frac{1}{8}\log\left(\frac{5}{4}\right)\right) - O(n\log n)\\
&> \frac{n^2}{25}.
\end{aligned}
$$

The sum of the memory requirement of $|A|$ routers of G is at least $n^2/25$ bits, that completes the proof. □

Theorem 2 states that *at least one* graph has a memory requirement of $\Omega(n^2)$ bits. Actually, it is easy to check, by a counting argument[6], that $(1-o(1))2^{n^2/25} \sim 2^{n^2/25}$ graphs need a total of $n^2/25$ bits to route with stretch factor $s<3$.

It is well-known that almost all the graphs have diameter 2. We strengthen Theorem 2 by showing that $\Omega(n^2)$ bits can be required for some worst-case graphs of diameter 2. The next lemma will be useful for the main theorem.

Lemma 2 *Let c be an integer, and let p,q be two sufficiently large integers, $0<c<q$. Let $\mathcal{M}$ be the set of $p\times q$ boolean matrices having c 1-entries per row. Let $\mathcal{M}_1$ be the subset of matrices of $\mathcal{M}$ such that all the columns are pairwise non complemented. Let $\mathcal{M}_2$ be the subset of matrices of $\mathcal{M}$ such that for every pair of rows the $2\times q$ matrix composed of the pair of rows contains the submatrix*[7] $\left[\begin{smallmatrix}0&0&1&1\\0&1&0&1\end{smallmatrix}\right]$ *up to a column permutation. Then,*

- *if $q=o(2^{p/2})$, then $|\mathcal{M}_1|\sim|\mathcal{M}|$;*
- *if $p=o(r^{q/2})$, then $|\mathcal{M}_2|\sim|\mathcal{M}|$, where $r=2q^{-1/(2q)}$ if $c=q/2$, and $r=\alpha^{-2\alpha}(2\alpha-1)^{2\alpha-1}$, $\alpha=\max\{c/q,1-c/q\}$ otherwise.*

Proof. We have $|\mathcal{M}|=\binom{q}{c}^p$. Let us compute an upper bound of the cardinality of the complement set $\overline{\mathcal{M}_1}$ of $\mathcal{M}_1$ in $\mathcal{M}$. $|\overline{\mathcal{M}_1}|=|\mathcal{M}|-|\mathcal{M}_1|$, that is the number of matrices that have at least two columns complemented. Every matrix of $\overline{\mathcal{M}_1}$ can be built as follows: choose a pair of columns, fill arbitrary one, complement the second one, then fill arbitrary the $q-2$ other columns with $c-1$ 1-entries

[6] Mainly because at least a fraction of $1-2^{-c}$ of all the binary strings of length K cannot be compressed by more than c bits. For example $c=\log K$ already gives that a fraction of $1-o(1)$ of all the K-strings have Kolmogorov Complexity $K-O(\log K)\sim K$ bits.

[7] A is a submatrix of B if A can be obtained from B by removing some columns and rows in B.

(therefore rows have a total of exactly c 1-entries). The number of ways to do it is:

$$|\overline{\mathcal{M}_1}| \leq \binom{q}{2} 2^p \binom{q-2}{c-1}^p.$$

To show that $|\mathcal{M}_1| \sim |\mathcal{M}|$, it is sufficient to show that $|\overline{\mathcal{M}_1}|/|\mathcal{M}| = o(1)$.

$$\frac{|\overline{\mathcal{M}_1}|}{|\mathcal{M}|} \leq \binom{q}{2} \left(2\binom{q-2}{c-1} \Big/ \binom{q}{c} \right)^p = \binom{q}{2} \left(\frac{2c(q-c)}{q(q-1)} \right)^p.$$

Let us set $\alpha = c/q$. Thus,

$$\frac{|\overline{\mathcal{M}_1}|}{|\mathcal{M}|} \leq \binom{q}{2} \left(2\alpha(1-\alpha)\frac{q}{q-1} \right)^p.$$

By assumption $q = o(2^{p/2})$, hence $\binom{q}{2} = o(2^p)$. Moreover $2\alpha(1-\alpha)$ is upper-bounded by $1/2$, and $(q/(q-1))^p = (1 - 1/(q-1))^p < e^{-(q-1)p} = O(1)$. It follows that:

$$\frac{|\overline{\mathcal{M}_1}|}{|\mathcal{M}|} \leq o(2^p) \left(\frac{1}{2}\right)^p O(1) = o(1).$$

Similarly, let us compute $|\overline{\mathcal{M}_2}| = |\mathcal{M}| - |\mathcal{M}_2|$. Every matrix of $\overline{\mathcal{M}_2}$ can be obtained as follows. Choose a pair of rows, fill arbitrarily one row with c 1-entries and fill the second one, using one of the 3 following cases in order to fail the property: 1) copy all the 0-entries of the first row and fill arbitrarily the other positions such that there are c 1-entries in the second row – the case $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ is missing; 2) if $c \geq q/2$, complement all the 0-entries and fill arbitrarily the other positions so that the second row has c 1-entries – the case $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$ is missing; 3) if $c \leq q/2$, complement all the 1-entries and fill arbitrarily the other positions so that the second row has c 1-entries – the case $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$ is missing. Finally, fill arbitrarily the $p-2$ remaining rows with c 1-entries per row.

Indeed, cases 1), 2) and 3) cover all possibilities. Concerning case 1), we observe that, after copying the 0-entries, the c remaining entries must be chosen equal to 1 in order to fulfill the requirement on the number of 1-entries per row. Thus, both rows are equal and correspond to the unique possibility to avoid both $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$. In case 2), we have $(q-c)$ 1-entries after complementing the 0-entries. So, we must complete the row in an arbitrary way, selecting $(2c-q)$ 1-entries, $2c - q \geq 0$, in the remaining c entries. So, we avoid $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$. Similarly, if $c \leq q/2$, case 3) avoids $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$.

In a first step, let us assume that $c \geq q/2$. The number of ways to select members of $\overline{\mathcal{M}_2}$, if $c \geq q/2$, is at most:

$$|\overline{\mathcal{M}_2}| \leq \binom{p}{2}\binom{q}{c}\left(1 + \binom{c}{2c-q}\right)\binom{q}{c}^{p-2}.$$

Note that the case $c = q/2$ is trivial, because it gives

$$\frac{|\overline{\mathcal{M}_2}|}{|\mathcal{M}|} \leq 2\binom{p}{2} \Big/ \binom{q}{q/2}.$$

Since $p = o((2q^{-1/(2q)})^{q/2}) = o(2^{q/2}/q^{1/4})$, hence $2\binom{p}{2} = o(2^q/\sqrt{q})$. Moreover $\binom{q}{q/2} = \Theta(2^q/\sqrt{q})$, hence

$$\frac{|\overline{\mathcal{M}_2}|}{|\mathcal{M}|} \leq o\left(\frac{2^q}{\sqrt{q}}\right) \Big/ \Theta\left(\frac{2^q}{\sqrt{q}}\right) = o(1).$$

So, we can assume is the following that $q/2 < c < q$. Because $\binom{c}{2c-q} = \binom{c}{q-c} \geq 1$, and $|\mathcal{M}| = \binom{q}{c}^p$, it follows:

$$\frac{|\overline{\mathcal{M}_2}|}{|\mathcal{M}|} \leq 2\binom{p}{2}\binom{c}{q-c} \Big/ \binom{q}{c}. \tag{3}$$

Let $\alpha = c/q$, $1/2 < \alpha < 1$.

$$\binom{c}{q-c} \Big/ \binom{q}{c} = \binom{\alpha q}{(1-\alpha)q} \Big/ \binom{q}{\alpha q}.$$

From Equation 1 of Lemma 1,

$$\binom{q}{\alpha q} > \frac{H(\alpha)^q}{\sqrt{2\pi\alpha(1-\alpha)q}} \quad \text{and} \quad \binom{\alpha q}{(1-\alpha)q} < \beta \frac{H\left(\frac{1-\alpha}{\alpha}\right)^{\alpha q}}{\sqrt{2\pi\left(\frac{1-\alpha}{\alpha}\right)\left(1-\frac{1-\alpha}{\alpha}\right)\alpha q}}.$$

After simplification,

$$\binom{c}{q-c} \Big/ \binom{q}{c} < \beta \cdot f(\alpha) \cdot w(\alpha)^q,$$

with

$$f(\alpha) = \frac{\alpha}{\sqrt{2\alpha - 1}} \quad \text{and} \quad w(\alpha) = \frac{H\left(\frac{1-\alpha}{\alpha}\right)^{\alpha}}{H(\alpha)}.$$

We have

$$f'(\alpha) = \frac{1}{\sqrt{2\alpha-1}}\left(1 - \frac{\alpha}{2\alpha - 1}\right).$$

Because in the range $]1/2, 1[$, $\alpha/(2\alpha - 1) < 1/2$, $f'(\alpha) \geq 0$, hence $f(\alpha) < f(1) = 1/\sqrt{2}$. It follows

$$\frac{|\overline{\mathcal{M}_2}|}{|\mathcal{M}|} < \frac{2\beta}{\sqrt{2}}\binom{p}{2} w(\alpha)^q.$$

It remains to compute $w(\alpha)^q$.

$$w(\alpha) = \frac{\left(\left(\frac{\alpha}{1-\alpha}\right)^{(1-\alpha)/\alpha}\left(\frac{\alpha}{2\alpha-1}\right)^{(2\alpha-1)/\alpha}\right)^{\alpha}}{\left(\frac{1}{\alpha}\right)^{\alpha}\left(\frac{1}{1-\alpha}\right)^{1-\alpha}} = \alpha^{2\alpha}(2\alpha-1)^{1-2\alpha} = r^{-1}.$$

By assumption $p = o(r^{q/2})$, hence $(2\beta/\sqrt{2})\binom{p}{2} = o(r^q)$. Therefore,

$$\frac{|\overline{\mathcal{M}_2}|}{|\mathcal{M}|} < o(r^q)\, r^{-q} = o(1).$$

Now, if $c < q/2$, it suffices in Inequality 3 to replace c by $q-c$, i.e., to consider the 0-entries instead of the 1-entries. Clearly, the property of matrices of $\mathcal{M}_2$ is symmetric for the 0-entries and the 1-entries.

$$\frac{|\overline{\mathcal{M}_2}|}{|\mathcal{M}|} \leq 2\binom{p}{2}\binom{q-c}{c} \Big/ \binom{q}{q-c}.$$

Choosing $\alpha = 1 - c/q$, we have $q - c = \alpha q$, and $c = (1-\alpha)q$. It turns out:

$$\frac{|\overline{\mathcal{M}_2}|}{|\mathcal{M}|} \leq 2\binom{p}{2}\binom{\alpha q}{(1-\alpha)q} \Big/ \binom{q}{\alpha q}.$$

We remark that $1/2 < \alpha < 1$, thus by the previous case

$$\frac{|\overline{\mathcal{M}_2}|}{|\mathcal{M}|} < o(r^q)\, w(\alpha)^q = o(1).$$

Both cases can be merged if we set $\alpha = \max\{c/q, 1 - c/q\}$, that completes the proof. $\square$

We can state now the main result of the paper.

Theorem 3 *For every sufficiently large integer n, there exists an n-node graph of diameter 2 on which every routing algorithm of stretch factor $s < 3$ has a total memory requirement of $n^2/25$ bits at least.*

Proof. Let us consider the set $\mathcal{M}' = \mathcal{M}_1 \cap \mathcal{M}_2$ introduced in Lemma 2, and let us show that all the graphs built from $\mathcal{M}'$ are of diameter 2.

Let $M \in \mathcal{M}'$ and let $G = (V, E)$ be its corresponding graph, constructed as in Theorem 2. The distance between any a_i (or b_i) and any v_j is obviously at most 2, see Figure 3. $M \in \mathcal{M}_1$ implies that columns of M are pairwise non complemented. Thus for every j, j' there exists an i such that $m_{i,j} = m_{i,j'}$. If $m_{i,j} = m_{i,j'} = 1$, then v_j and $v_{j'}$ are both neighbors of a_i. If $m_{i,j} = m_{i,j'} = 0$, both are neighbors of b_i. This implies that $d_G(v_j, v_{j'}) = 2$.

Since $M \in \mathcal{M}_2$, M has the property that, for any two rows i, i' with $i \neq i'$, there exist j_1 such that $m_{i,j_1} = 1 = m_{i',j_1}$, j_2 such that $m_{i,j_2} = 0 = m_{i',j_2}$, j_3 such that $m_{i,j_3} = 1 \neq m_{i',j_3}$ and j_4 such that $m_{i,j_4} = 0 \neq m_{i',j_4}$. Thus, $d_G(a_i, a_{i'}) = 2$ as $m_{i,j_1} = 1 = m_{i',j_1}$ implies that v_{j_1} is common neighbor of both a_i and $a_{i'}$, for j_1 correctly chosen. The other cases are proved in the same way, whence $d_G(a_i, a_{i'}) = d_G(b_i, b_{i'}) = d_G(a_i, b_{i'}) = d_G(b_i, a_{i'}) = 2$ and G of diameter 2.

Let us choose $p \sim n/4$, and $q \sim n/2$, as we did in Theorem 2. Let $c = \alpha q$, and $\alpha = 4/5$. From Lemma 2, $|\mathcal{M}'| \sim |\mathcal{M}|$, if $q = o(2^{p/2})$, and $p = o(r^{q/2})$, with $r = (5/16) \cdot 4^{2/5} \cdot 3^{3/5}$. We get $r \approx 1.05$, $q = o(2^{n/8})$, and $p = o(1.05^{n/4})$, thus $|\mathcal{M}'| = (1 - o(1))|\mathcal{M}|$. Let us apply the proof of Theorem 2 for the set $\mathcal{M}'$ instead of $\mathcal{M}$. Equation 2 gives for a graph built from $\mathcal{M}'$:

$$\sum_{x \in A} \mathrm{MR}(x) \ > \ \log \left(|\mathcal{M}'| \Big/ \binom{2(q-c)}{q-c}^p \right) - O(n \log n).$$

with $|\mathcal{M}'| = (1 - o(1)) \binom{q}{c}^p$. For $\alpha = c/q = 4/5$, Lemma 1 gives:

$$|\mathcal{M}'| \Big/ \binom{2q/5}{q/5}^p \ > \ (1 - o(1)) \gamma^p \left(\frac{5}{4} \right)^{pq}.$$

Thus,

$$\begin{aligned} \sum_{x \in A} \mathrm{MR}(x) &> \log \left((1 - o(1)) \gamma^p \left(\frac{5}{4} \right)^{pq} \right) - O(n \log n) \qquad (4) \\ &> pq \log \left(\frac{5}{4} \right) - O(p) - O(n \log n) \quad (\gamma \text{ constant} < 1) \\ &> n^2 \left(\frac{1}{8} \log \left(\frac{5}{4} \right) \right) - O(n \log n) \\ &> \frac{n^2}{25}. \end{aligned}$$

□

4 Improving the Lower Bounds

From Theorem 3 it is straightforward to conclude that some routers can require $0.04\,n$ bits at least. We remark that we can say better.

Theorem 4 *For every sufficiently large integer n, there exists an n-node graph of diameter 2 on which every routing algorithm of stretch factor $s < 3$ has a local memory requirement asymptotically of $n \log(5/4) \approx 0.32\, n$ bits at least.*

Proof. Consider the same graphs built than in Theorem 3 from the set of matrices $\mathcal{M}'$, but with $p = \Theta(n/\log n)$, and $q = n - 2p \sim n$. Let choose again $\alpha = c/q = 4/5$. To apply Lemma 2 it suffices to check that $q = o(2^{\Theta(n/\log n)})$, and $p = o(1.05^{n/2-o(n)})$. So, $|\mathcal{M}'| \sim |\mathcal{M}|$. From Inequality 4 of Theorem 3, there exists at least one router of A, x_0, such that:

$$\mathrm{MR}(x_0) \ > q \log\left(\frac{5}{4}\right) - O\left(\frac{n \log n}{p}\right) \quad \text{(recall that } |A| = p\text{)}.$$

Which implies that $\mathrm{MR}(x_0) > n \log(5/4) - O(\log^2 n)$. $\square$

David Peleg posted in [11] the question whether there is a way to design compact *k-regional routing schemes* for all graphs, that is a routing algorithm that routes messages optimally up to distance k. As we saw in Introduction $n + O(\log n)$ bits are sufficient to implement in each router a 1-regional routing scheme. Theorem 4 shows that 1-regional routing schemes on arbitrary n-node graphs cannot be designed with less than $n \log(5/4)$ bits in all the routers. So, it would be interesting to determine the exact bound.

Actually the bound of the total memory requirement in Theorem 2 can be improved by a factor two, but the graph is not of diameter two.

Theorem 5 *For every sufficiently large integer n, there exists an n-node graph of diameter 3 on which every routing algorithm of stretch factor $s < 3$ has a total memory requirement $n^2/13$ bits at least.*

Proof. Let G be any graph built like in Theorem 2 from a $p \times q$ matrices of a suitable subset $\mathcal{M}'' \subset \mathcal{M}$. Choose $c = 4q/5$, $p = \lfloor n/2 \rfloor$, and $q = \lceil n/2 \rceil$. We transform G into G'' obtained by removing all the nodes b_i of G. G'' has $p+q = n$ nodes exactly. Let us define $\mathcal{M}''$, $M \in \mathcal{M}''$ if and only if any two rows have at least a 1-entry in common on the same column, and if any tow columns have at least a 1-entry in common in the same row. We check that for $p \sim n/2$, $q \sim n/2$, and $c \sim 2n/5$, $|\mathcal{M}''| = (1 - o(1))|\mathcal{M}|$ (the proof is quite similar that to show in Lemma 2 $|\mathcal{M}_1| \sim |\mathcal{M}|$). Moreover, G'' has a diameter 3. Indeed, $d_{G''}(a_i, a_{i'}) = 2$ (there exists a column j such that $m_{i,j} = m_{i',j} = 1$); $d_{G''}(v_j, v_{j'}) = 2$ (there exists a row i such that $m_{i,j} = m_{i,j'} = 1$); however $d_{G''}(a_i, v_j) = 1$ or 3, depending if v_j is a neighbor of a_i or not.

Let R be any routing algorithm on G'' of stretch factor $s < 3$. The proof of Theorem 2 applies for G'' as well: if a router a_i would not routes to all its neighbors optimally, R would be of stretch factor $s \geq 3$ (the girth of G'' is still 4). The rebuilding procedure of Theorem 2 does not depend of the nodes b_i. So M can be rebuilt from the local routing functions of the routers a_i of G'' with the same procedure than the one used to rebuilt M from the router a_i of G. In this case Inequality 4 holds:

$$\begin{aligned}\sum_{x \in A} \mathrm{MR}(x) &> pq \log\left(\frac{5}{4}\right) - O(n \log n) \\ &> n^2 \left(\frac{1}{4} \log\left(\frac{5}{4}\right)\right) - O(n \log n) \\ &> \frac{n^2}{13}.\end{aligned}$$

□

5 Conclusion

We proved a lower bound of $\Omega(n^2)$ bits for the entire network to route with any stretch factor $s < 3$. Let $s(m)$, for every integer function m, be the smallest real such that every n-node graph supports a distributed algorithm of stretch factor $s \leq s(m)$ with a total of $o(m)$ bits. We showed in this paper that there is a gap for stretch factor 3, and $s(n^2) = 3$.

For a theoretical point of view it would be interesting to compute the constant $s(n^2 \log n)$, which corresponds to the threshold stretch factor for compression of routing tables. At the present time, best estimations are $1 < s(n^2 \log n) \leq 3$ (see [5] for the lower bound).

The same evaluation on stretch factor threshold for compression of routing information, $\overline{s}(m)$, could be done for the average of all the networks, rather than for the worst-case. Works of [2] tends to show a quite different behavior between $s(m)$ and $\overline{s}(m)$, for instance $\overline{s}(n^2 \log n) = 1$.

Acknowledgments

We would like to thank André Raspaud and Guy Mélançon for their help in the proof of Lemma 2.

References

[1] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, *Improved routing strategies with succint tables*, Journal of Algorithms, 11 (1990), pp. 307–341.

[2] H. Buhrman, J.-H. Hoepman, and P. Vitányi, *Optimal routing tables*, in 15^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), May 1996.

[3] P. Flocchini, B. Mans, and N. Santoro, *Sense of direction: Formal definitions and properties*, in 1^{st} International Colloquium on Structural Information & Communication Complexity (SIROCCO), P. Flocchini, B. Mans, and N. Santoro, eds., Carleton University Press, May 1994.

[4] P. Fraigniaud and C. Gavoille, *Memory requirement for universal routing schemes*, in 14^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM PRESS, ed., Aug. 1995, pp. 223–230.

[5] C. Gavoille and S. Pérennès, *Memory requirement for routing in distributed networks*, in 15^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM PRESS, ed., May 1996, pp. 125–133.

[6] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics, a foundation for computer science (second edition)*, Addison-Wesley, 1994.

[7] E. Kranakis and D. Krizanc, *Lower bounds for compact routing*, in 13^{th} Annual Symposium on Theoretical Aspects of Computer Science (STACS), C. Puech and R. Reischuk, eds., vol. 1046 of Lecture Notes in Computer Science, Springer-Verlag, Feb. 1996, pp. 529–540.

[8] E. Kranakis, D. Krizanc, and J. Urrutia, *Compact routing and shortest path information*, in 2^{nd} International Colloquium on Structural Information & Communication Complexity (SIROCCO), L. M. Kirousis and E. Kranakis, eds., Carleton University Press, June 1995, pp. 101–112.

[9] M. Li and P. M. B. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, 1993.

[10] D. Peleg and E. Upfal, *A trade-off between space and efficiency for routing tables*, Journal of the ACM, 36 (1989), pp. 510–530.

[11] Siena Research School, *Communication at the Siena Research School '97 on "Compact Routing and Sense of Direction"*, June 1997.

[12] E. M. Wright, *Graphs on unlabelled nodes with a given number of edges*, Acta Math., 126 (1971), pp. 1–9.