Local Memory Requirement of Universal Routing Schemes

Pierre Fraigniaud and Cyril Gavoille *

Laboratoire de l'Informatique du Parallélisme - CNRS
École Normale Supérieure de Lyon
69364 Lyon cedex 07
France

Abstract

In this paper, we deal with the compact routing problem, that is the problem of implementing routing schemes that use a minimum memory size on each router. A *universal* routing scheme is a scheme that applies to all networks. In [13], Peleg and Upfal showed that one can not implement a universal routing scheme with less than a total of $\Omega(n^{1+1/(2s+4)})$ memory bits for any routing scheme satisfying that the maximum ratio between the lengths of the routing paths and the lengths of the shortest paths, that is the *stretch factor*, is bounded by s. In [6], Fraigniaud and Gavoille improve this bound by proving that universal routing schemes of stretch factors at most 2 use a total of $\Omega(n^2)$ memory bits in the worst-case. Recently, Gavoille and Pérennès [9] showed that, in fact, in the worst-case, $\Theta(n)$ routers of an n-node network may require up to $\Omega(n \log n)$ memory bits for shortest path routing. In this paper, we extend this result by showing that, for any constant ε, $0 < \varepsilon < 1$, $\Omega(n^\varepsilon)$ routers of an n-node network may require up to $\Omega(n \log n)$ memory bits even if each routing path is of length up to twice the distance between its source and its destination.

1 Introduction and statement of the problem

One of the most important measures of complexity of routing schemes is the size of the routing information that must be stored locally and globally in a network. In this paper, we use a similar model as the one introduced by Peleg and Upfal in [13]. A *routing function* R is a triple (I, H, P) consisting of *initialization*, *header*, and *port* functions, respectively. For any two distinct nodes u and v, R produces a path $u = u_0, u_1, \ldots, u_k = v$ of nodes, and a sequence $h_0, h_1, \ldots, h_k$ of headers such that $h_0 = I(u, v)$, $P(u_k, h_k) = \emptyset$, and for all i, $1 \leq i < k$, $H(u_i, h_i) = h_{i+1}$, $P(u_i, h_i) = (u_i, u_{i+1})$. We are interested in the distributed way of implementing routing functions. In this paper, we assume that nodes are labeled by integers in $\{1, \ldots, n\}$, and that the output ports of each node x are labeled by integers in $\{1, \ldots, \deg(x)\}$, where n and $\deg(x)$ are the total number of nodes in the network and the degree of the node x, respectively.

*Both authors are supported by the research programs ANM and PRS of the CNRS, and by the DRET of the DGA.

A *routing scheme* is a function that returns a routing function R for any network G. For instance, the *shortest path interval routing scheme* [14, 15] is the selection, for any network G, of a shortest path routing function R whose implementation is based on grouping in intervals the destination addresses associated to each arc, and whose aim is to minimize the number of intervals per link. For any network, the obtained routing function R may require a large number of intervals but it exists. The shortest path interval routing scheme is therefore said to be *universal* because it applies to all networks. On the contrary, the shortest path 1-interval routing scheme, that is the routing scheme that returns a shortest path routing function that can be implemented using a unique interval per link, is only *partial* in the sense that there exist networks for which no such routing scheme exists [5].

We consider the standard model of point-to-point communication networks described as finite connected symmetric digraphs $G = (V, E)$. In the following, we denote by n the number of vertices of such a graph. The vertices represent the processors or the nodes of the network, and the edges represent bidirectional communication links between the nodes (to each edge corresponds two symmetric arcs). A vertex can directly communicate with its neighbors only, and messages between nonadjacent vertices are sent along a path connecting them in the network. In the following, we always refer to *finite connected symmetric digraph* by the simple term *graph*.

Let G be any graph, and let x be any vertex of G. For every routing function R on G, we denote by $\mathrm{MEM}(G, R, x)$ the minimum *memory requirement* of R in x. The parameter $\mathrm{MEM}(G, R, x)$ is simply the Kolmogorov complexity [10] of the local computation of R in x. Of course, the memory requirement is defined for a fixed coding strategy. From the definition above, the memory requirement does not take into account the size of the header. Indeed, to be as general as possible, we allow headers to be of unbounded size. We then define, for any routing function R on a graph G:

- the *global memory requirement* as

$$\mathrm{MEM}_{global}(G,R) = \sum_x \mathrm{MEM}(G,R,x),$$

- the *local memory requirement* as

$$\mathrm{MEM}_{local}(G,R) = \max_x \mathrm{MEM}(G,R,x).$$

The *stretch factor* of a routing function R on a graph G is denoted by $s(R,G)$, and satisfies

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

where $d_G(x,y)$ denotes the distance between x and y in G, and $d_R(x,y)$ denotes the length of the routing path between x and y constructed by R. For every stretch factor s, the *global memory requirement* of G is $\mathrm{MEM}_{global}(G,s) = \min_R \mathrm{MEM}_{global}(G,R)$, and the *local memory requirement* of G is $\mathrm{MEM}_{local}(G,s) = \min_R \mathrm{MEM}_{local}(G,R)$, where the minima are taken over all the routing functions R on G of stretch factor at most s.

It is interesting to consider both global and local memory requirements because it is possible that $\mathrm{MEM}_{global}(G,s) \ll n\,\mathrm{MEM}_{local}(G,s)$ for some graph G of n vertices, and for some stretch factor s. That is, informally, the global memory requirement does not indicate whether the routing information can be evenly balanced between all the routers. On the contrary, the local memory requirement does not indicate whether all the routers need such a large memory.

For example, $\mathrm{MEM}_{local}(H_n,1) = O(\log n)$, where H_n is the hypercube of order n, by using e-cube routing [3]. It is known that, for all acyclic graphs [14], for all outer-planar graphs [7], and for all unit circular-arc graphs [5] G of order n and of maximum degree d, $\mathrm{MEM}_{local}(G,1) = O(d \log n)$ because the 1-interval routing scheme applies to these graphs (1 interval per arc). For every chordal graph G, it is shown in [11] and [13] that $\mathrm{MEM}_{global}(G,3) = O(n \log^2 n)$. Let us consider the complete graph K_n of order n. In general, in a vertex x, a local routing function R in x on K_n is stored with $\Theta(n \log n)$ bits for a random labeling of the ports of x, or a port labeling chosen by an adversary. Indeed, an adversary can choose some permutation π between the $n-1$ port-labels and the $n-1$ neighbors of x in such a way that reaching any neighbor of x implies to know the full description of π, *i.e.* $\lceil \log_2((n-1)!) \rceil = \Theta(n \log n)$ bits. However, it is easy to see that some other routing functions (using suitable port labeling) can be described locally with $O(\log n)$ bits. Therefore, we have $\mathrm{MEM}_{local}(K_n,1) = O(\log n)$.

The following table summarizes the state of the art on the local and global memory requirements, and the best known memory complexity of universal routing schemes on networks of order n as a function of the stretch factor s. When no reference is indicated, routing tables are the best known routing scheme.

Comments.
In the referenced papers, the bounds on the memory requirement of the routing schemes are always given in term of $n^{1/k}$ where $k \geq 1$ is an integer function raising with the stretch factor. Table 1 presents results in an other way so that one can compare them as a function of the stretch factor.

- In [1], the stretch factor is at most $2^k - 1$, and the routing scheme allows non uniform cost on the arcs. Another routing scheme is proposed in [1]. It locally requires at most $O(k(d + n^{1/k}) \log n)$ memory bits on any vertex of degree d for every stretch factor at most $2 \cdot 3^k - 1$. Therefore, this routing scheme is efficient in the case of bounded degree networks.
- In [2], the stretch factor is at most $16k^2$, and the routing scheme allows non uniform cost on the arcs. The bound on the local memory requirement is given as a function of the diameter of the network: $O(kn^{1/k} \log n \log D)$ bits per router. In order to compare all the results, we consider the worst-case, that is $D = O(n)$. Headers, and vertex labeling are of size $O(\log n)$ bits. Note that $n^{1/\sqrt{\log n}} = e^{\sqrt{\log n}}$.
- In [13], the stretch factor is at most $12k + 3$. However, the routing scheme does not support non uniform cost on the arc. Moreover, the routing scheme proposed in this paper is not evenly balanced on all the routers. This is why we can not use this routing scheme for the upper bound on the local memory requirement. Headers are of size $O(\log n)$ bits, but this routing scheme uses a vertex labeling on $O(\log^2 n)$ bits. For large value of the stretch factor, this scheme gives a tight upper bound on the global memory requirement.

As one can see in the table, for a stretch factor $s < 16$, and for the local memory requirement, the best known universal routing scheme is the one based on routing tables. It is of first interest to know whether this bound is tight because, in practice, near-shortest path routing schemes are the most important. We will show that, indeed, this bound is tight for any stretch factor $s < 2$.

Theorem 1 *For any stretch factor s, $1 \leq s < 2$, for any constant ε, $0 < \varepsilon < 1$, and for any n, there exists an n-node network G_n in which $\Omega(n^\varepsilon)$ routers require $\Theta(n \log n)$ bits each to code any routing function of stretch at most s.*

Therefore, in Table 1, the entry "$1 < s < 2$" must be replaced by

stretch	local memory req.	global memory req.
$1 < s < 2$	$\mathbf{\Theta(n \log n)}$	$\Omega(n^2)$ [6] $O(n^2 \log n)$

To prove this result, we will extend the tools introduced in [4, 8]. In particular, we will define the *generalized matrices of constraints* (Section 2), and the *generalized graphs of constraints* (Section 3). Section 4 gives the proof of Theorem 1. Finally, Section 5 contains some concluding remarks, and open problems.

2 Generalized Matrices of Constraints

In this section, we introduce a more general and powerful definition of *matrix of constraints* [4, 8] to improve the lower bound on the local memory requirement of universal routing

stretch	local memory requirement	global memory requirement
$s=1$	$\Theta(n \log n)$ [9]	$\Theta(n^2 \log n)$ [9]
$1<s<2$	$\Omega(n)^\dagger$ [6] $O(n \log n)$	$\Omega(n^2)$ [6] $O(n^2 \log n)$
$2 \leq s < 3$	$\Omega(n^{1/(2s+4)})^\dagger$ [13] $O(n \log n)$	$\Omega(n^{1+1/(2s+4)})$ [13] $O(n^2 \log n)$
$3 \leq s < 16$	$\Omega(n^{1/(2s+4)})^\dagger$ [13] $O(n \log n)$	$\Omega(n^{1+1/(2s+4)})$ [13] $O(n^{1+1/\lfloor \log_2(s+1) \rfloor} \log^2 n)$ [1]
$16 \leq s < 87$	$\Omega(n^{1/(2s+4)})^\dagger$ [13] $O(n^{1/\lfloor \sqrt{s}/4 \rfloor} \log^2 n)$ [2]	$\Omega(n^{1+1/(2s+4)})$ [13] $O(n^{1+1/\lfloor \log_2(s+1) \rfloor} \log^2 n)$ [1]
$s \geq 87$	$\Omega(n^{1/(2s+4)})^\dagger$ [13] $O(\sqrt{s} n^{1/\lfloor \sqrt{s}/4 \rfloor} \log^2 n)$ [2]	$\Omega(n^{1+1/(2s+4)})$ [13] $O(s^3 n^{1+1/\lfloor (s-3)/12 \rfloor} \log n)$ [13]
$s = O(\log n)$	$\Omega(1)^\dagger$ [13] $O(e^{\sqrt{\log n}} \log^{3/2} n)$ [2]	$\Omega(n)$ [13] $O(n \log^4 n)$ [13]
$s = O(\sqrt{n})$	$\Omega(1)^\dagger$ [13] $O(e^{\sqrt{\log n}} \log^{3/2} n)$ [2]	$\Omega(n)$ [13] $O(n \log n)$ [12]

†Bounds derived from the lower bound of the global memory requirement.

Table 1: Best known bounds on the memory requirement.

schemes for stretch factor $s<2$. We will specify the routing constraints using the label of the output port that a vertex must use to send a message. More formally, we define a generalized matrix of constraints as follows.

Definition 1 *Let $G=(V,E)$ be a graph, and s be any real ≥ 1. A* generalized matrix of constraints *of G of stretch* factor *s is a $p \times q$ integer matrix $M=(m_{i,j})$, $i \in \{1,\dots,p\}$ and $j \in \{1,\dots,q\}$, such that the entries of the row i of M are in $\{1,\dots,|\cup_{1 \leq j \leq q} \{m_{i,j}\}|\}$, $i \in \{1,\dots,p\}$. Moreover, there exist two sets of vertices $A=\{a_i \mid 1 \leq i \leq p\} \subset V$ and $B=\{b_j \mid 1 \leq j \leq q\} \subset V$, and there exist p functions $\psi_i : \{1,\dots,|\cup_{1 \leq j \leq q}\{m_{i,j}\}|\} \to E$, $i \in \{1,\dots,p\}$, satisfying: for every routing function $R=(I,H,P)$ on G of stretch factor at most s, and for every (i,j), $1 \leq i \leq p$, $1 \leq j \leq q$,*

$$\psi_i(m_{i,j}) = P(a_i, I(a_i,b_j)).$$

Informally, if $M=(m_{i,j})$ is a generalized matrix of constraints of a graph G, Definition 1 means that there exist two subsets of vertices A and B such that every near-shortest path between $a_i \in A$ to $b_j \in B$ starts with the arc $e_{i,j}$ which is locally labeled by the integer $m_{i,j}$. In the following, we call *constrained vertices* the vertices of the set A, and *target vertices* the vertices of the set B. The matrices of constraints defined in [4, 8] can be seen as a restricted case of the generalized matrices of constraints defined above when the degree is 2 and $s=1$. Subsequently, generalized matrices of constraints are called matrices of constraints for short.

Figure 1 shows a matrix of constraints of shortest path of the Petersen-graph, with the constrained vertices $A=\{a_1,\dots,a_5\}$, and the target vertices $B=\{b_1,\dots,b_5\}$. For example, on this graph, every shortest path from vertex a_3 to vertex b_2 has to start with the arc (a_3,b_1). It means that it is possible to fix the labels of the incident arcs of the vertices of A such that every shortest path routing function on the Petersen-graph uses port 1 to send a message from a_3 to b_2.

If M is a matrix of constraints of a graph G, then, from Definition 1, any matrix obtained from M by permutation of rows and columns, and by permutation of entries of each row is also a matrix of constraints of G. It simply corresponds to another labeling of the vertices and arcs of G. Of course, vertex and arc labeling of G have significant implications on the size of the coding of a routing function R on G (see the example of the complete graph in Section 1 on the page 2). By hypothesis, we are looking for the best labeling so that we obtain the most compact coding of R.

Definition 2 *Let $M=(m_{i,j})$, and $M'=(m'_{i,j})$ be two $p \times q$ integer matrices, $i \in I=\{1,\dots,p\}$ and $j \in J=\{1,\dots,q\}$. We set $M \sim M'$ if and only if there exist two permutations r of I and c of J, and p permutations ψ_i of $\{1,\dots,|\cup_{j \in J}\{m_{i,j}\}|\}$, $i \in I$, such that, for every $i \in I$, and for every $j \in J$:*

$$m'_{i,j} = \psi_i(m_{r(i),c(j)}).$$

The relation $\sim$ is an equivalence relation because r, c and ψ_i, $i \in \{1,\dots,p\}$, are bijective functions. $M \sim M'$ means that M' can be obtained from M by permutation of the rows and of the columns, and by permutation of the entries of each rows. Permutations ψ_i are related with arc labeling. Permutations c and r are related with vertex labeling (of constrained and target vertices).

We can define a canonical representative of each equivalence class of integer matrices. In the following, we choose as canonical representative of a set of $p \times q$ integer matrices

↱	b_1	b_2	b_3	b_4	b_5
a_1	1	1	2	2	3
a_2	1	1	2	2	3
a_3	1	1	2	2	3
a_4	1	1	2	2	3
a_5	1	2	2	1	3

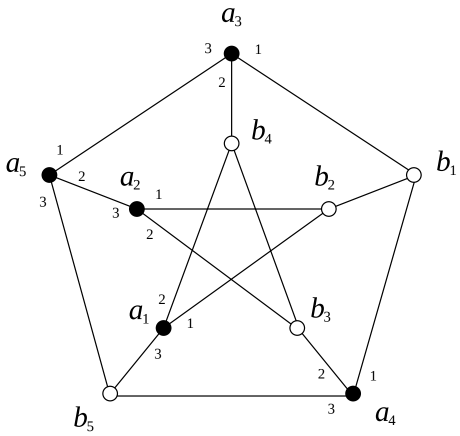


Figure 1: An example of matrix of constraints of shortest path on the Petersen-graph.

$M = (m_{i,j})$ the matrix $M_0 \sim M$ that minimizes the integer $\sum_{i,j} m_{i,j} q^{(p-i)q+(q-j)}$. This integer is called the *index* of the matrix. It can be understood as the value of the integer $l_1 l_2 \dots l_p$, where l_i denotes the i-th row of M, represented in base q.

Notation. For every triple of integers (p,q,d), we denote by d-$\mathcal{M}_{p,q}$ the set of canonical representatives of the different equivalence classes of $\sim$ for the $p \times q$ integer matrices with entries in $\{1,\dots,d\}$. For example 3-$\mathcal{M}_{2,3}$ is the set:

$$3\text{-}\mathcal{M}_{2,3} = \left\{ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 2 \\ 1 & 1 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix} \right\} \quad (1)$$

This example shows that every 2×3 integer matrices with entries from $\{1,2,3\}$ is equivalent to one of the matrices of 3-$\mathcal{M}_{2,3}$. For example matrix $\left[\begin{smallmatrix} 1 & 2 & 2 \\ 1 & 3 & 2 \end{smallmatrix}\right]$ is of index 479, and is equivalent (under $\sim$) to the matrix $\left[\begin{smallmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \end{smallmatrix}\right] \in$ 3-$\mathcal{M}_{2,3}$ which is of minimum index 396.

Lemma 1 *For every triple of integers (p,q,d),*

$$\frac{d^{pq}}{p!\,q!\,(d!)^p} \le |d\text{-}\mathcal{M}_{p,q}|. \textit{Therefore,}$$

$$\log_2 |d\text{-}\mathcal{M}_{p,q}| = \Omega((pq - pd)\log d - q \log q - p \log p).$$

Proof. There are d^{pq} $p \times q$ integer matrices with entries in $\{1,\dots,d\}$. At most $p!q!$ matrices are equivalent under row and column permutation, and, for each row, there are at most $d!$ distinct results obtained by permutation of entries inside the row (therefore at most $(d!)^p$ permutations for all the rows). Hence there are at least $d^{pq}/(p!q!(d!)^p)$ distinct classes of integer matrices. □

3 Generalized Graphs of Constraints

We will see that, for every integer matrix M, there exists a graph, called *generalized graph of constraints*, that has M as matrix of constraints.

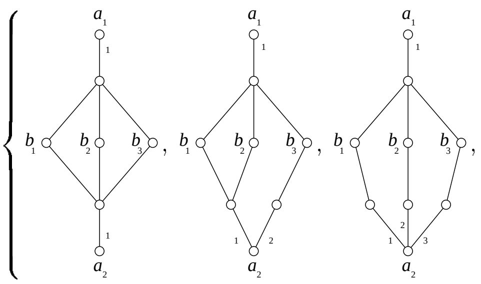
Lemma 2 *For every matrix $M \in d$-$\mathcal{M}_{p,q}$, there exists a graph G of order at most $p(d+1)+q$ such that M is a matrix of constraints of G of stretch factor < 2.*

Proof. Let (p,q,d) be a triple of integers, and $M = (m_{i,j})$ be any matrix in d-$\mathcal{M}_{p,q}$. We construct a graph $G = (V,E)$ composed of three distinct levels as follows: $V = A \cup B \cup C$, with $A = \{a_1,\dots,a_p\}$, $B = \{b_1,\dots,b_q\}$, and $C \subset \{c_{i,k} \mid 1 \le i \le p \text{ and } 1 \le k \le d\}$. Vertices are connected as follows: for every $a_i \in A$, $\{a_i, c_{i,k}\} \in E$ if and only if there exists j such that $m_{i,j} = k$, and $\{b_j, c_{i,k}\} \in E$ if and only if $m_{i,j} = k$. No other arc belongs to E. Isolated vertices of C are removed.

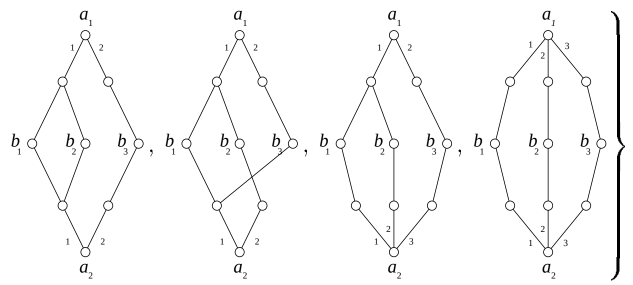
The local port labeling of vertex a_i is: $(a_i, c_{i,k})$ is labeled k. One can check that M is a matrix of constraints of G where A is the set of constrained vertices, and B is the set of target vertices. Moreover, if $m_{i,j} = k$, then there exits a unique path of length 2 from a_i to b_j: the path $a_i, c_{i,k}, b_j$. Moreover, the other paths from a_i to b_j that do not start with arc $(a_i, c_{i,k})$ are of length at least 4. Therefore, M is a matrix of constraints of stretch factor < 2 in G. The order of G is at most $p(d+1)+q$, because $|A| = p$, $|B| = q$ and $|C| \le pd$. □

The graphs constructed following the rules given in the proof of Lemma 2 are called *generalized graphs of constraints* (*graphs of constraints* for short). For every triple of integers (p,q,d), we denote by d-$\mathcal{G}_{p,q}$ the set of graphs of constraints of all the matrices in d-$\mathcal{M}_{p,q}$. Equation 2 shows the set 3-$\mathcal{G}_{2,3}$.

3-$\mathcal{G}_{2,3}$ =



(2)



4 Proof of Theorem 1

To prove Theorem 1, we will use a "local" argument that is, for any matrix $M \in d\text{-}\mathcal{M}_{p,q}$, the routers of the constrained vertices of a graph of constraints of the matrix M are able to rebuild the matrix M.

Let (p, q, d) be a triple of integers. For every matrix $M \in d\text{-}\mathcal{M}_{p,q}$, let G be its graph of constraints, and let n denote its order. Let $A = \{a_1, \dots, a_p\}$ be the constrained vertices, and $B = \{b_1, \dots, b_q\}$ be the target vertices. Let R be a routing function on G such that $s(R, G) < 2$. R uses particular vertex and arc labeling. Without loss of generality, one can assume that the output ports of vertices a_i are labeled by the $m_{i,j}$ (otherwise consider another matrix M' equivalent to M). By definition, every communication between $a_i \in A$, and $b_j \in B$ is sent out from a_i by the output port labeled $m_{i,j}$. Let M_B be the number of bits necessary to code the list of the labels of the vertices of B. Let M_C be the number of bits necessary to describe a function that computes the canonical representative of any $p \times q$ integer matrix with entries from $\{1, \dots, d\}$. Clearly, since we can do this reasoning for every matrix $M \in d\text{-}\mathcal{M}_{p,q}$, and since there exists at least one matrix $M \in d\text{-}\mathcal{M}_{p,q}$ that requires at least $\log_2 |d\text{-}\mathcal{M}_{p,q}|$ bits to be coded, we get that, for this matrix:

$$\sum_{a \in A} \mathrm{MEM}(G, R, a) + M_B + M_C + O(\log n) \geq \log_2 |d\text{-}\mathcal{M}_{p,q}|.$$

Indeed, to rebuild M, it is sufficient to test all routers of the vertices in A on all the labels of the target vertices, and to store the results in a matrix M'. To do that, it is enough to know the routing functions R in a, $a \in A$, the labels of the vertices in B, and a way to find the canonical representative of the equivalence class of the matrix M' obtained. To simplify the process, one can even assume that the integers p, q and d are given on $O(\log n)$ bits.

Since the vertex-labels are in $\{1, \dots, n\}$, the list of labels of B can be described with $M_B = \log_2 \binom{n}{q} + O(\log n)$ bits. Indeed, there are $\binom{n}{q}$ ways of choosing q labels among the n possible choices. Moreover $M_C = O(\log n)$ because a simple algorithm depending of p, q, and d can compute all the permutations of rows, columns, and entries of each rows of M in order to minimize the index. Since, for every $q \in \{1, \dots, n\}$, $\log_2 \binom{n}{q} \leq n = O(pd + q)$ (see Lemma 2), it follows that

$$\sum_{a \in A} \mathrm{MEM}(G, R, a) = \Omega((pq - pd) \log d - q \log q - p \log p) \quad (3)$$

by applying Lemma 1.

Let ε be any given constant, $0 < \varepsilon < 1$, and let n be large enough. Let $p = \lfloor n^{\varepsilon}/2 \rfloor$, $d = \lfloor n^{1-\varepsilon} \rfloor - 1$, and $q = \lfloor n/2 \rfloor$. For every matrix $M \in d\text{-}\mathcal{M}_{p,q}$, applying Lemma 2, we obtain a graph of constraints G' of M of order $n' \leq p(d+1) + q \leq n$. We can transform G' in a graph G by adding a path of $n - n'$ vertices to a vertex that is not a constrained vertex nor a target vertex. Clearly, M is still a matrix of constraints with stretch factor < 2 of G, and G is of order n. Therefore, by Equation 3,

$$\sum_{a \in A} \mathrm{MEM}(G, R, a) = \Omega((1 - \varepsilon) n^{1+\varepsilon} \log n) = \Omega(n^{1+\varepsilon} \log n). \quad (4)$$

Therefore, since $|A| = p = \Omega(n^{\varepsilon})$, there exists at least one router $a_0 \in A$ that requires $\Omega(n \log n)$ bits to code any routing function R on G of stretch < 2. Note that $O(n \log n)$ bits are sufficient using routing tables. Therefore, at least $\Omega(n^{\varepsilon})$ routers require $\Theta(n \log n)$ bits each to code any routing function on G of stretch factor < 2. This completes the proof of Theorem 1.

5 Conclusion

Theorem 1 implies that routing tables can not be locally compressed (asymptotically) in the worst-case. This completes the study of the local memory requirement of universal routing schemes of stretch factor $s < 2$.

- An interesting question would be to ask whether we can close the gap between the bound of $\Omega(n^2)$ bits [6] and the upper bound of $O(n^2 \log n)$ bits (using routing tables) for the global memory requirement of routing functions of stretch factor $s < 2$.
- Another interesting question would be to determine the smallest stretch factor for which routing tables can be locally compressed asymptotically, that is computing the largest stretch factor s_0 for which there exist graphs of order n that require $\Omega(n \log n)$ memory bits on at least one of their routers. Table 1 shows that $s_0 \leq 16$, implying that routing schemes of stretch factor greater than 16 can locally compress routing tables.
- As one can see on Table 1, large stretch factors allow to strongly compress the routing information. Is it possible to reduce the upper bounds on the local memory requirement in the same way as it was done for the global memory requirement?

Acknowledgments. Both authors are thankful to Stéphane Pérennès for his valuable comments for the proof of Theorem 1.

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] B. Awerbuch and D. Peleg, *Routing with polynomial communication-space trade-off*, SIAM Journal on Discrete Mathematics, 5 (1992), pp. 151–162.

[3] W. J. Dally and C. L. Seitz, *Deadlock-free message routing in multiprocessor interconnection networks*, IEEE Transaction on Computers, C-36 (1987), pp. 547–553.

[4] M. Flammini, J. van Leeuwen, and A. M. Spaccamela, *Lower bounds on interval routing*, Tech. Rep. 69, Università di L'Aquila, Dipartimento di matematica Pura ed Applicata, Oct. 1994.

[5] P. Fraigniaud and C. Gavoille, *Optimal interval routing*, in Parallel Processing: CONPAR '94 - VAPP VI, B. Buchberger and J. Volkert, eds., vol. 854 of Lecture Notes in Computer Science, Springer-Verlag, Sept. 1994, pp. 785–796.

[6] ———, *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.

[7] G. N. Frederickson and R. Janardan, *Designing networks with compact routing tables*, Algorithmica, 3 (1988), pp. 171–190.

[8] C. Gavoille and E. Guévremont, *Worst case bounds for shortest path interval routing*, Research Report 95-02, LIP, École Normale Supérieure de Lyon, 69364 Lyon Cedex 07, France, Jan. 1995.

[9] 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. To appear.

[10] A. N. Kolmogorov, *Three approaches to the quantitative definition of information*, Problems Inform. Transmission, 1 (1965), pp. 1–7.

[11] D. Peleg and A. A. Schäffer, *Graph spanners*, Journal of Graph Theory, 13 (1989), pp. 99–116.

[12] D. Peleg and E. Upfal, *Efficient message passing using succint routing tables*, Research Report RJ 5768, IBM, 1987.

[13] ———, *A trade-off between space and efficiency for routing tables*, in 20^{th} Annual ACM Symposium on Theory of Computing (STOC), Chicago, IL, May 1988, pp. 43–52.

[14] N. Santoro and R. Khatib, *Labelling and implicit routing in networks*, The Computer Journal, 28 (1985), pp. 5–8.

[15] J. van Leeuwen and R. B. Tan, *Interval routing*, The Computer Journal, 30 (1987), pp. 298–307.