# Memory Requirement for Universal Routing Schemes

Pierre Fraigniaud and Cyril Gavoille
{pfraign, gavoille}@lip.ens-lyon.fr
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 implementing routing schemes that use a minimum memory size on each router. In [20], Peleg and Upfal showed that there is no hope to do that with less than a total $\Omega(n^{1+1/(2s+4)})$ memory bits for any stretch factor $s \geq 1$. We improve this bound for stretch factors $s < 2$ by proving that any near-shortest path routing scheme uses a total of $\Omega(n^2)$ memory bits.

## 1 Introduction

The *general routing problem* in a network (as opposed to the *permutation routing problem* [18] or the *broadcasting problem* [8]) consists of finding a routing *protocol* or routing *function* such that, for any source-destination pair, any message from the source can be routed to the destination. $XY$-routing [2] or $e$-cube routing [3] are such protocols. The efficiency of a protocol is measured in terms of latency (related to the length of the paths) and/or throughput (related to link or node congestion). Finding shortest paths in a network is easy [14], and finding paths minimizing congestion can be done easily for a large class of networks (namely Cayley graphs) [16]. However, the routings protocols that are obtained may be of no use from a VLSI point of view. Indeed, the implementation of these protocols may require such a large amount of hardware that they cannot be used in practice.

For instance, implementing $XY$ or $e$-cube routing is simple because routing is locally performed by respectively comparison and XOR-ing of the local address with the destination address. However, networks do not necessarily have the simple structure of meshes or hypercubes. For instance routing in *pancake* networks requires to decompose any given permutation in a product of particular ones, and this is definitely not easy (see Problem 39 in [9]). Also, networks may even have no structure at all so that a routing algorithm cannot be clearly defined on them. In these cases, *routing tables* are generally used: when a node receives a message, it consults its local table to determine the output port of the message. Such a process is fast, that is the latency of each router is small. However each router needs to store a large table of size $\Theta(n \log \Delta)$ for a regular graph of order $n$ and degree $\Delta$. Moreover, since routers must be replicated $n$ times in an $n$-node network, such a large local memory requirement implies a global information of $\Theta(n^2 \log \Delta)$ bits for the whole network, just for routing.

Many authors propose methods to compress the routing table (see for instance [21, 11, 12]). Among them, *interval routing* [21, 22, 1, 6] was chosen by *Inmos* for its C104 routing chip [4]. Unfortunately, it was proven in [17, 5, 6] that the number of intervals per link necessary to route in any network is not bounded when using shortest path routing (actually it was recently proven in [13] that this number is in $\Theta(n)$). Since the C104 routing chip only allows a few intervals per link, it means that there exist networks such that no shortest path routing protocol on them can be implemented using C104 chips. In this sense, this chip is not "universal", that is it cannot be used for all possible networks.

To formalize this concept, let us define a *routing scheme* to be a function that returns a routing protocol $R$ for any network $G$. For instance, the *shortest path interval routing scheme* is the selection, for any network $G$, of a shortest path routing function $R$ whose implementation minimizes 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 on which this routing scheme is not defined [7].

The universality of a routing scheme is counterbalanced by the large memory that may be required the routers to implement the routing functions of this scheme. For instance, it is proved in [13] that the shortest path interval routing scheme requires a router with $\Theta(n \log n)$ memory bits to be implemented on any network. Partial routing schemes require less memory (for instance $\Theta(\log n)$ memory bits for the shortest path 1-interval routing scheme), but they do not apply on all networks. In the last Colloquium on Structural Information and Communication Complexity, Danny Krizanc asked the informal question "what is the minimum size of a universal routing scheme providing routing functions satisfying some given constraint as shortest paths ?". In other words, how much and in what situation is it possible to compress the routing tables? This problem is addressed in this paper.

In fact, Peleg and Upfal partially answered Krizanc's question in [20]. They showed that any shortest path universal routing scheme uses a total of $\Omega(n^{7/6})$ memory bits. Therefore we can derive from their global lower bound that any shortest path universal routing scheme requires routers with at least $\Omega(\sqrt[6]{n})$ memory bits. In this paper, we improve the result of Peleg and Upfal by showing that any shortest path universal routing scheme uses a total of $\Omega(n^2)$ memory bits, that is a shortest path universal router needs at least a memory of $\Omega(n)$ bits. This shows that, in some sense, tables are only up to a logarithmic factor worst than the best possible universal router when shortest paths are considered.

To be more precise, we not only consider shortest path routing but also routing for which the ratio length of the routing paths over length of shortest paths is bounded by a constant $s$ called the *stretch factor*. We show that any routing scheme of stretch $s < 2$ uses a total of $\Omega(n^2)$ memory bits. Hence, for small stretches, our result improves the lower bound of Peleg and Upfal that is $\Omega(n^{1+1/(2s+4)})$ for any stretch factor $s \geq 1$ [20]. Our lower bound also implies that the local memory requirement for a router must be $\Omega(n)$ bits.

In the next section we describe the model of routing functions considered in this paper. Section 3 introduces definitions for *global* and *local memory requirement* of a routing function, and states the main results. Proofs of these results are given in Section 4.

## 2 A Model of Routing Function

We consider the standard model of a point-to-point communication network, described by a connected symmetric digraph $G = (V, E)$. 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 opposite arcs). A vertex can communicate directly only with its neighbors, and messages between nonadjacent vertices are sent along a path connecting them in the network. We denote by $\deg(x)$ the degree of the vertex $x$ and by $d_G(x, y)$ the distance from $x$ to $y$ in $G$. In the following we always refer to *connected symmetric digraph* by the simple term *graph*.

Also we can generalize the model by considering networks with nonnegative weights on the arcs. Such a model can be important for practical applications to model differences between the links of the network: different link latencies or different link bandwidths for example. To establish a lower bound on the memory requirement, we should consider the worst weight assignment on the worst case graph. In this paper, the networks are constructed with a particular weight assignment: a unit-cost on all the arcs. Therefore even if our bound might be non optimal, it applies to any weighted graph.

We use the general definition of routing function given in [20].

**Definition 1 (Routing Function)** *Let $G = (V, E)$ be any graph and $\mathcal{H}$ be any set called header set. A* routing function *on $G$ is a triple $R = (I, H, P)$ of functions such that:*

- *$I$ is the* initialization *function, $I : V \times V \to \mathcal{H}$*
- *$H$ is the* header *function, $H : V \times \mathcal{H} \to \mathcal{H}$*
- *$P$ is the* port *function, $P : V \times \mathcal{H} \to E$*

*Moreover, for every pair $(x, y)$ of vertices of $G$, $x \neq y$, there exists a family of $k+1$ vertices $\{v_i \mid v_i \in V, 0 \leq i \leq k\}$ and a family of $k+1$ headers $\{h_i \mid h_i \in \mathcal{H}, 0 \leq i \leq k\}$ such that $v_0 = x$, $v_k = y$, $h_0 = I(x, y)$ and for every $i \in \{0, \ldots, k-1\}$, $H(v_i, h_i) = h_{i+1}$ and $P(v_i, h_i) = (v_i, v_{i+1})$.*

In other words, if $R$ is a routing function on a graph $G$, the path built by $R$ is as follows. If the vertex $x$ wants to send a message $M$ to the vertex $y$ in $G$, it prepares a header $h$ for the message $M$ by computing $h = H(x, I(x, y))$ and attaches it to the message. Then it selects the port $p$ by computing $p = P(x, h)$ and sends the message onto this port. Now, suppose that the message $M$ with a header $h'$ arrives at some vertex $z$. The first step consists in reading $h'$ and in checking whether it is the final destination. If so, the routing process ends. Otherwise, $z$ must forward the message further by preparing a new header $h'' = H(z, h')$ and by replacing the old header $h'$ attached to the message by the new one, and by computing the port $p' = P(z, h'')$ and sends the message onto this port $p'$.

**Notation.** The distributed representation of $R$ is the collection $\{R_x = (I_x, H_x, P_x) \mid x \in V\}$ such that for every $x, y \in V$ and $h \in \mathcal{H}$, $I_x(y) = I(x, y)$, $H_x(h) = H(x, h)$ and $P_x(h) = P(x, h)$. For every $x \in V$, $R_x$ is called the local routing function of $R$ in $x$. The length of the path induced by $R$ from $x$ to $y$ is denoted by $d_R(x, y)$.

In this model, we consider oblivious routing functions, *i.e.* routing functions $R$ such that the path induced by $R$ from $x$ to $y$ only depends of $x$ and $y$. However the local header function $H_x$ used to compute the next header depends on the current header attached to the message. Similarly, a local port function $P_x$ also depends on this header. This allows to employ flexible *trial-and-error* routing functions. That is the paths induced by $R$ can be paths with loops. This property may allow to decrease the local memory requirement for the storage of information because it allows to visit many nodes to get more information. This implies that our bounds also apply for the usual routing functions $R = \{R_x = (Identity, Identity, P_x) \mid x \in V\}$ on a graph $G = (V, E)$ with $\mathcal{H} = V$, where headers always contain the name of the destination.

Note that our lower bound on the memory requirement can also be applied to the case of *adaptive* routing functions, since oblivious routing functions are particular cases of adaptive routing functions.

With these previous notations we define the *stretch factor* introduced in [19] as follows:

**Definition 2 (Stretch Factor)** *Let $R$ be any routing function on a graph $G$. The* stretch factor *of $R$ on $G$ is denoted by $s(R, G)$ and satisfies*

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

The stretch factor at least has two main interests. First, it measures well the impact of the routing, better than only considering the length of the longest path. But mainly the stretch factor measures the tradeoff between space (needs of collecting informations) and the time (length of the paths).

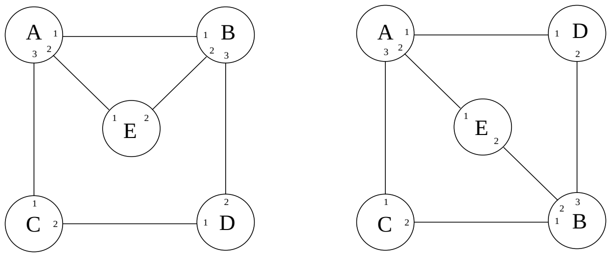
As it is said in [20], the same routing function can be used on several nonisomorphic graphs, because neighbors of a vertex can be specified by a local port function $P_x$ rather than a one-to-one labeling on the set of arcs. Indeed, it is not efficient for a local port function $P_x$ to return an absolute label of the output arc, it saves space to return a local label of this port between 1 and $\deg(x)$. The example on Figure 1, taken from [20], shows two *isomorphic* shortest path routing functions on two nonisomorphic graphs represented by a single one routing table (at the bottom of Figure 1). In this example $\mathcal{H} = V$.

Let us define more formally what are two isomorphic functions.

**Definition 3 (Isomorphic Routing Function)** *Let $R = (I, H, P)$ be any routing function on a graph $G$ with header set $\mathcal{H}$ and $R' = (I', H', P')$ be any routing function on a graph $G'$ with header set $\mathcal{H}'$. The function $R$ is* isomorphic *to $R'$ if and only if there exist three bijections $\varphi$, $\psi$ and $\Gamma$ where $\varphi : V \to V'$, $\psi : E \to E'$, $\Gamma : \mathcal{H} \to \mathcal{H}'$ that satisfy for every $x, y \in V$, $x \neq y$ and for every $h \in \mathcal{H}$:*

*(1)* $I'(\varphi(x), \varphi(y)) = \Gamma \circ I(x, y)$
*(2)* $H'(\varphi(x), \Gamma(h)) = \Gamma \circ H(x, h)$
*(3)* $P'(\varphi(x), \Gamma(h)) = \psi \circ P(x, h)$

Informally the vertices and the arcs of both graphs $G$ and $G'$ can be labeled in different ways, as so the headers of $\mathcal{H}$ and $\mathcal{H}'$ such that a same table as Figure 1 can be used for both routing functions.



| ↗ | A | B | C | D | E |
|---|---|---|---|---|---|
| A | - | 1 | 3 | 1 | 2 |
| B | 1 | - | 1 | 3 | 2 |
| C | 1 | 2 | - | 2 | 1 |
| D | 1 | 2 | 1 | - | 2 |
| E | 1 | 2 | 1 | 2 | - |

Figure 1: An example due to Peleg and Upfal of a same routing table that applies on two nonisomorphic graphs

Our model being defined, let us now state our results.

## 3 Memory Requirement

Krizanc's question is related to the following definition:

**Definition 4 (Memory Requirement of a Router)** *Let $G$ be any graph, let $x$ be any vertex of $G$ and let $s$ be any real. For every routing function $R$ on $G$ of stretch factor at most $s$, we denote by $\mathrm{MEM}(G, s, R, x)$ the minimum memory requirement to store the local function $R_x$.*

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 assume headers of unbounded size. Definition 4 allows us to define the local and global memory requirement as follows.

**Definition 5 (Global/Local Memory Requirement)** *Let $G$ be any graph, let $s$ be any real and let $R$ be any routing function on $G$ of stretch factor at most $s$. We define $\mathrm{MEM}_{global}(G, s, R) = \sum_x \mathrm{MEM}(G, s, R, x)$ and $\mathrm{MEM}_{local}(G, s, R) = \max_x \mathrm{MEM}(G, s, R, x)$. The* global memory requirement *of $G$ is $\mathrm{MEM}_{global}(G, s) = \min_R \mathrm{MEM}_{global}(G, s, R)$ and the* local memory requirement *of $G$ is $\mathrm{MEM}_{local}(G, s) = \min_R \mathrm{MEM}_{local}(G, s, R)$, overall 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 $s$. In this paper we focus our attention on the worst cases on $\text{MEM}_{global}(G,s)$ and $\text{MEM}_{local}(G,s)$, for small values of $s$. For every integer $n$, we define $\text{MEM}_{global}(n,s) = \max_G \text{MEM}_{global}(G,s)$ and $\text{MEM}_{local}(n,s) = \max_G \text{MEM}_{local}(G,s)$, overall the graphs $G$ of order $n$.

For example $\text{MEM}_{local}(H_n,1) \in^1 O(\log n)$, where $H_n$ is the hypercube of order $n$, by using $e$-cube routing. It is known that for all acyclic graphs [21], for all outerplanar graphs [10] and for all unit circular-arc graphs [7] $G$ of order $n$ and of maximum degree $\Delta$, $\text{MEM}_{local}(G,1) \in O(\Delta \log n)$, because the 1-interval routing scheme applies to these graphs (1 interval per arc).

Note that our model does not require to store the address of the vertex locally, unlike to the definition of memory requirement of Peleg and Upfal [20]. Consider for instance the complete graph $K_n$ of order $n$. We have $\text{MEM}_{local}(K_n,1) \in \Theta(1)$ because, for every vertex $x$, $I_x = H_x = P_x = Identity$ and thus the memory requirement to compute $R$ is constant. Note that is does not mean that the router itself will have a constant size. The memory requirement is only a lower bound on the *surface* of a router.

With the same model, Peleg and Upfal showed that for every stretch factor $s$, $\text{MEM}_{global}(n,s) \in \Omega(n^{1+1/(2s+4)})$ and $\text{MEM}_{local}(n,s) \in \Omega(\sqrt[2s+4]{n})$. Below is our contribution:

**Theorem 1** *For every $s < 2$, $\text{MEM}_{global}(n,s) \in \Omega(n^2)$.*

This directly implies the following corollary:

**Corollary 1** *For every $s < 2$, $\text{MEM}_{local}(n,s) \in \Omega(n)$.*

Of course, by using routing tables, we get that $\text{MEM}_{local}(n,1) \in O(n \log n)$. However, it seems that the distribution of $R$ in the form of local functions $R_x$ locally requires an overhead of $O(\log n)$ bits due to the redundancy of the information. In fact we conjecture the following:

**Conjecture 1** *$\text{MEM}_{local}(n,1) \in \Theta(n \log n)$.*

That conjecture would imply that tables cannot be compressed in the worst case.

## 4 Proof of Theorem 1

Our proof is based on the concept of routing scheme.

**Definition 6 (Routing Scheme)** *Let $n$ be any integer, let $\mathcal{G}$ be any set of graphs of order $n$, and let $\mathcal{R}$ be any set of routing functions. A* routing scheme *is a function $RS : \mathcal{G} \to \mathcal{R}$ such that, for every graphs $G \in \mathcal{G}$, $RS(G)$ is a routing function of $\mathcal{R}$ on $G$. Moreover $RS$ is said to be a* universal routing scheme *if $\mathcal{G}$ is the set of all the graphs of order $n$.*

Table routing is such a universal routing scheme, but, not a 1-interval routing scheme.

### 4.1 Sketch of the proof

We show that for every $n \geq 20$, any universal routing scheme $RS : \mathcal{G} \to \mathcal{R}$, where $\mathcal{R}$ is the set of routing functions of stretch factor $s < 2$, constructs $2^{\Theta(n^2)}$ nonisomorphic routing functions. To prove this, we consider a large class $\mathcal{G}_n$ of graphs of $\mathcal{G}$ such that two nonisomorphic routing functions of $\mathcal{R}$ correspond to two different graphs of $\mathcal{G}_n$.

To build the class $\mathcal{G}_n$, composed of the so called *graphs of constraints*, we extend the notion of *matrix of constraints* introduce in [13, 5]. Each matrix of constraints of a graph represents a set of constraints on the paths required by any routing function to be of stretch factor $s$.

### 4.2 Matrices of Constraints

**Definition 7 (Matrix of Constraints)**
*Let $G = (V,E)$ be a graph, $V' \subset V$ and $E' \subset E$. Let $s$ be any real. A* matrix of constraints *of $E'$ on $V'$ of stretch $s$ in $G$ is a $|V'| \times |E'|$ boolean matrix $M = (m_{v,e})$, $v \in V', e \in E'$, such that the rows of $M$ are labeled by the vertices of $V'$ and the columns of $M$ are labeled by the arcs of $E'$, and such that: $\forall v \in V', \forall e = (x,y) \in E'$*

$m_{v,e} = 1 \Leftrightarrow$ *all paths of length less or equal to $s\, d_G(x,v)$ from $x$ to $v$ in $G$ use the arc $e$,*

*and*

$m_{v,e} = 0 \Leftrightarrow$ *no path of length less or equal to $s\, d_G(x,v)$ from $x$ to $v$ in $G$ uses the arc $e$.*

Given a graph $G$ and a $p \times q$ boolean matrix of constraints $M$ in $G$, then every other matrix that belongs to the equivalence class of $p \times q$ boolean matrices, using row and column permutation as a congruence operator, is also a matrix of constraints in $G$.

**Definition 8 (Index, Canonical Representative)**
*Let $M = (m_{i,j})$, $1 \leq i \leq p, 1 \leq j \leq q$, be a $p \times q$ boolean matrix. The* index *of $M$ is the positive integer*

$$\mu(M) = \sum_{1 \leq i \leq p, 1 \leq j \leq q} m_{i,j} 2^{(p-i)+p(q-j)}$$

*For any equivalence class $C$ of $p \times q$ boolean matrices, the* canonical representative *of $C$ is the unique $p \times q$ boolean matrix $M_0 \in C$ such that $\mu(M_0) = \max_{M \in C} \mu(M)$.*

[1] In this paper $O(f)$ (resp. $\Omega(f)$) is the class of functions $g$ such that $\exists n_0 \in \mathbb{N}, \exists \alpha \in \mathbb{R} \mid \forall n \geq n_0$, $g(n) \leq \alpha f(n)$ (resp. $g(n) \geq \alpha f(n)$). $\Theta(f) = O(f) \cap \Omega(f)$.

The index $\mu(M)$ of a $p \times q$ boolean matrix $M$ can be seen as the value of the binary expression of $M$ obtained by the concatenation of $q$ words (the columns of $M$) of $p$ bits.

For every pair of integers $(p, q)$, $\mathcal{M}_{p,q}$ denotes the set of $p \times q$ boolean matrices $M$ such that $M$ is the canonical representative of the equivalence classes of matrices under row permutations, column permutations and/or bit complementation, and without column containing exactly one 1-entry or one 0-entry. For example:

$$\mathcal{M}_{4,2} = \left\{ \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \right\} \quad (1)$$

**Lemma 1** *For every pair of integers $(p, q)$, $(2^p - p)^q/(p!q!2^q) \leq |\mathcal{M}_{p,q}| \in 2^{\Theta(pq)}$.*

**Proof.** There are $(2^p - p)^q$ matrices with no column with a single 1-entry. Therefore there are at least $(2^p - p)^q/(p!q!)$ classes of matrices with no single 1-entry on each column that are different even up to a permutation of rows and columns. Now, there are a most $2^q$ possible bit complementations of the $q$ columns. For more details about the number of boolean matrices classes, see [15]. □

Let $\sim$ be the equivalence relation under row and column permutation for the set of boolean matrices. In the following, if $A$ and $B$ are two $p \times q$ boolean matrices, then we denote by $(AB)$ the concatenation of the matrices $A$ and $B$, that is a $p \times 2q$ boolean matrix.

**Lemma 2** *Let $M, M' \in \mathcal{M}_{p,q}$, for any pair of integers $(p, q)$. If $M \neq M'$ then $(M\overline{M}) \not\sim (M'\overline{M'})$.*

**Proof.** Let $(p, q)$ be any pair of integers and $M, M' \in \mathcal{M}_{p,q}$ such that $M \neq M'$. By definition of the set $\mathcal{M}_{p,q}$, $M \neq M' \Rightarrow M \not\sim M'$, thus $\mu(M) = \max_{X \sim M} \mu(X)$ and $\mu(M') = \max_{X \sim M'} \mu(X)$. Let us show that

$$\max_{X \sim (M\overline{M})} \mu(X) = \mu(MY) \quad (2)$$

for a suitable $p \times q$ boolean matrix $Y$. Indeed, assume by contradiction that $\max_{X \sim (M\overline{M})} \mu(X) = \mu(NZ)$ with $N \neq M$. Thus there exist a permutation $r$ of rows of $(M\overline{M})$ and a permutation $c$ of columns, such that $NZ = r(c(M\overline{M}))$ and $\mu(NZ) > \mu(MY)$, for any matrix $Y$ obtained from $\overline{M}$ by its row and column permutations, and where $r(X)$ (resp. $c(X)$) denotes the matrix obtained by permutation of rows (resp. columns) of the matrix $X$. In the following we denote by $m_i$ a column of $M$, for $i \in \{1, \ldots, q\}$, and by $\overline{m_i}$ the corresponding column $i$ in the matrix $\overline{M}$. Since $N \neq M$ then $N$ has a column $\overline{m_j}$. Moreover the column $m_j$ can not belong to the matrix $Z$, because $M \in \mathcal{M}_{p,q}$ and the matrix obtained from $M$ by complementing the column $j$ and applying the row permutation $r$ would give a greater index than $\mu(M)$. Thus $N$ contains both columns $m_j$ and $\overline{m_j}$, and therefore the matrix $Z$ contains a column $m_k$ from $M$. We have $\mu(m_j) > \mu(m_k)$ (each column is considered as a $p \times 1$ matrix), otherwise the matrix obtained from $(NZ)$ by exchanging columns $m_j$ and $m_k$ would give a greater index than $\mu(NZ)$. Similarly $\mu(\overline{m_j}) > \mu(m_k)$. Since $\mu(M)$ is maximum under row, column and column bit complementation, it follows that $\mu(m_k) > \mu(\overline{m_k})$. Thus $\mu(\overline{m_j}) > \mu(m_k) > \mu(\overline{m_k})$, *i.e.* $\mu(\overline{m_j}) > \mu(\overline{m_k})$. That is a contradiction with $\mu(m_j) > \mu(m_k)$. We conclude that, up to a permutation $c$ of columns, all columns of $N$ are columns of $M$. Now it is easy to see that $r$ is necessary the identity function, otherwise $\mu(M)$ would be not maximum. Thus $N = M$ and we get Equation 2. Similarly $\max_{X \sim (M'\overline{M'})} \mu(X) = \mu(M'Y')$, for a suitable $p \times q$ boolean matrix $Y'$, and $\mu(MY) \neq \mu(M'Y')$ implies that $\mu(MY) \neq \mu(M'Y')$. Hence $\max_{X \sim (M'\overline{M'})} \mu(X) \neq \max_{X \sim (M\overline{M})} \mu(X)$, *i.e.* $(M\overline{M}) \not\sim (M'\overline{M'})$. □

### 4.3 Graphs of Constraints

We shall establish a bijective correspondence between the set of matrices of constraints and a particular set of graphs. For any symmetric digraph $G = (V, E)$, we denote by $\langle a, b \rangle$ the pair of arcs $\{(a, b), (b, a)\} \subset E$.

**Definition 9** *Let $M = (m_{i,j}), 1 \leq i \leq p, 1 \leq j \leq q$ ,be any $p \times q$ boolean matrix. The* graph of constraints *of the matrix $M$ is a symmetric digraph $G = (V, E)$ such that:*

- $V = \{a_1, \ldots, a_q\} \cup \{b_1, \ldots, b_q\} \cup \{c_1, \ldots, c_q\} \cup \{v_1, \ldots, v_p\}$
- $\langle c_j, a_j \rangle$ *and* $\langle c_j, b_j \rangle \in E$, *for every* $j \in \{1, \ldots, q\}$
- $\langle a_j, v_i \rangle \in E$ *if and only if* $m_{i,j} = 0$, *for every* $i \in \{1, \ldots, p\}$ *and* $j \in \{1, \ldots, q\}$
- $\langle b_j, v_i \rangle \in E$ *if and only if* $m_{i,j} = 1$, *for every* $i \in \{1, \ldots, p\}$ *and* $j \in \{1, \ldots, q\}$
- *no more arc belongs to $E$*

Equation 3 below shows four examples of graphs of constraints (see the set denoted $\mathcal{G}_{4,2}$), one for each matrix of the set $\mathcal{M}_{4,2}$ of Equation 1.

**Lemma 3** *Let $M$ be any $p \times q$ boolean matrix, and let $G = (V, E)$ be its graph of constraints. Then $M$ is a matrix of constraints of stretch factor $< 2$ in $G$. Moreover*

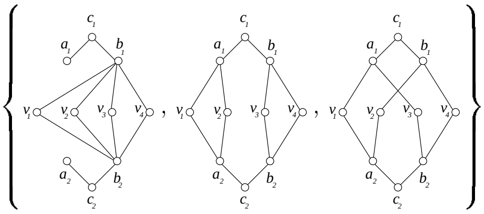
- $|V| = p + 3q$
- *For every* $j \in \{1, \ldots, q\}, \deg(a_j) + \deg(b_j) = p + 2$ *and* $\deg(c_j) = 2$
- *For every* $i \in \{1, \ldots, p\}, \deg(v_i) = q$

**Proof.** Consider the graph of constraints $G$ of a $p \times q$ boolean matrix $M$. Let $A$ be the set of arcs $(c_j, b_j)$, for

$j \in \{1, \dots, q\}$, and $B = \{v_1, \dots, v_p\}$. Clearly, by definition of $G$, $M$ is a matrix of constraints of $A$ on $B$ in $G$ of stretch factor $< 2$. Indeed, every arc $e \in A$ belongs to chordless cycle of length 6 and there exists a unique arc $e \in A$ such that $d_G(c_j, v_i) = 2$. Thus if a routing function $R$ on $G$ uses arc $e$ when it should not or does not use $e$ when it should, then $d_R(c_j, v_i) \geq d_G(c_j, v_i) + 2$ and therefore the stretch factor of $R$ will be at least $(2+2)/2 \geq 2$. □

For any pair of integers $(p, q)$, we denote by $\mathcal{G}_{p,q}$ the set of graphs of constraints of all the matrices of $\mathcal{M}_{p,q}$. Equation 3 gives the set $\mathcal{G}_{4,2}$ where the graphs are represented undirected.

$\mathcal{G}_{4,2} =$



(3)

The set $\mathcal{G}_{p,q}$ contains graphs that are pairwise nonisomorphic. For the purpose of our proof, we need something stronger, that is they do not support isomorphic routing functions of stretch factor $< 2$. This is done in the next section.

### 4.4 Enumeration of the Range of *RS*

For every function $f : A \to B$, we denote by $f(A)$ the range of $f$, *i.e.* $f(A) = \{y \in B \mid \exists x \in A, y = f(x)\}$.

**Theorem 2** *Let $\mathcal{R}$ be the set of routing functions of stretch factor $< 2$ on the graphs of order $n$. For any universal routing scheme $RS : \mathcal{G} \to \mathcal{R}$, $|RS(\mathcal{G})| \in 2^{\Theta(n^2)}$.*

**Proof.** Let *RS* be any universal routing scheme of $\mathcal{G}$ on $\mathcal{R}$ of routing functions of stretch factor $< 2$. Consider, for every integer $n \geq 20$, the subset of graphs $\mathcal{G}_n \subset \mathcal{G}$ such that $\mathcal{G}_n = \mathcal{G}_{p,q}$ where $p = \lfloor n/4 \rfloor - 3 + ((n \bmod 4) \bmod 3)$ and $q = p + 4\lfloor (n+5)/4 \rfloor - n$. Since the order of the graphs in $\mathcal{G}_{p,q}$ is $p + 3q$ (Lemma 3), then clearly, for every $n \geq 20$, the order of the graphs in the set $\mathcal{G}_n$ is exactly $n$. Let $M, M' \in \mathcal{M}_{p,q}$, $M \neq M'$. Let $G = (V, E)$ and $G' = (V', E')$ be the two corresponding graphs of constraints in $\mathcal{G}_n$. For every routing function $R = (I, H, P)$ on $G$ of header set $\mathcal{H}$ and for every routing function $R' = (I', H', P')$ of header set $\mathcal{H}'$ on $G'$, we will show that if $s(R, G) < 2$ and $s(R', G') < 2$, then $R$ is nonisomorphic to $R'$. We partition the set of vertices $V$ of $G$ in four subsets $A$, $B$, $C$ and $D$, grouping respectively the vertices $a_j$'s, $b_j$'s, $c_j$'s and $v_i$'s of $G$, for $j \in \{1, \dots, q\}$ and $i \in \{1, \dots, p\}$ (see Definition 9). We obtain $A'$, $B'$, $C'$ and $D'$ in partitioning the same for the sets of vertices $V'$ in the same way.

Assume that $R$ and $R'$ are isomorphic. Then there exist three bijections $\varphi$, $\psi$ and $\Gamma$ such that for every $x, y \in V, x \neq y$ and for every $h \in \mathcal{H}$, Equations (1), (2) and (3) of Definition 3 are satisfied. Let $x \in V$ and $x' = \varphi(x) \in V'$. Let us show that $x$ and $x'$ must have the same degree: assume without loss of generality that $\deg(x) > \deg(x')$. From Equation (3) of Definition 3, this implies that an arc $(x, y)$ of $E$ is not used by $R$. Now, graphs of constraints are without loops and multi-arcs, therefore the path from $x$ to $y$ in $G$ induced by $R$ is of length at least 2: a contradiction with the fact that $s(R, G) < 2$. Thus if $R$ and $R'$ are isomorphic, then $x$ and $\varphi(x)$ have the same degree, for every $x \in V$.

From Lemma 3 and by definition of the parameters $p$ and $q$ as functions of $n$, since $\lfloor n/4 \rfloor - 3 + ((n \bmod 4) \bmod 3) \geq 2$ and $4\lfloor (n+5)/4 \rfloor - n \geq 2$, for any $n \geq 20$, we get that $\deg(c) < \deg(t) < deg(v)$, for any vertices $c \in C \cup C'$, $v \in D \cup D'$ and $t \in A \cup B \cup A' \cup B'$. Therefore $\varphi(C) = C'$, $\varphi(D) = D'$ and $\varphi(A \cup B) = A' \cup B'$. Let $Z = \{(c, t) \in E \mid c \in C \text{ and } t \in A \cup B\}$ be the set of arcs of $E$ between vertices of $C$ and vertices of $A \cup B$, and let $Z' = \{(c, t) \in E' \mid c \in C' \text{ and } t \in A' \cup B'\}$.

Since $\varphi(C) = C'$, we get from equation (3) of Definition 3 that $\psi(Z) = Z'$. Indeed, for every $c \in C$ and for every $h \in \mathcal{H}$, $P(c, h) \in Z$ and $P'(\varphi(c), \Gamma(h)) \in Z'$. Let $N = (M\overline{M})$ be the $p \times 2q$ boolean matrix composed of the matrix $M$ and its bit complement matrix $\overline{M}$. Let $N' = (M'\overline{M'})$ be the $p \times 2q$ boolean matrix obtained similarly from $M'$. Matrix $N$ and $N'$ are respectively a matrix of constraints of $Z$ on $D$ in $G$, and a matrix of constraints of $Z'$ on $D'$ in $G'$.

Let $v \in D, e = (c, t) \in Z$ and $h_{c \to v} = I(c, v)$. The header $h_{c \to v}$ is the header generated by the routing function $R$ to send a message from $c$ to $v$ in $G$. $\Gamma(h_{c \to v}) = \Gamma \circ I(c, v) = I'(\varphi(c), \varphi(v))$ from Equation (1) of Definition 3. Thus $\Gamma(h_{c \to v}) = h_{\varphi(c) \to \varphi(v)}$ is the header generated by the routing function $R'$ to send a message from $\varphi(c)$ to $\varphi(v)$ in $G'$. Now, from the definition of $N$, for every $v \in D$, for every $e = (c, t) \in Z$

$$N_{v,e} = 1 \Leftrightarrow P(c, h_{c \to v}) = e.$$

This implies that

$$N_{v,e} = 1 \Leftrightarrow P'(\varphi(c), h_{\varphi(c) \to \varphi(v)}) = \psi(e).$$

Thus

$$N_{v,e} = 1 \Leftrightarrow N'_{\varphi(v), \psi(e)} = 1.$$

Hence $N$ and $N'$ are equivalent under row and column permutation (the ones generated by $\varphi$ and $\psi$ on $D$ and $Z$ respectively). On the other hand, we know that $M \neq M'$ (they are two different canonical representatives of two different equivalence classes of $\mathcal{M}_{p,q}$). Therefore, applying Lemma 2, $(M\overline{M}) \not\sim (M'\overline{M'})$, where $\sim$ is the equivalence relation under row and column permutation. This is a contradiction with $N \sim N'$. Hence $R$ and $R'$ cannot be isomorphic.

Hence $|\mathcal{G}_n| \leq |RS(\mathcal{G}_n)| \leq |RS(\mathcal{G})| \leq |\mathcal{G}|$ and therefore $|\mathcal{M}_{p,q}| \leq |RS(\mathcal{G})| \leq 2^{n(n-1)/2} \in 2^{\Theta(n^2)}$. From Lemma 1, $|\mathcal{M}_{p,q}| \in 2^{\Theta(pq)}$. Since $p \in \Theta(n)$ and $q \in \Theta(n)$ then $|RS(\mathcal{G})| \in 2^{\Theta(n^2)}$. □

**Remark.** If we would not have been interested in the problem of *distributed* routing schemes, Theorem 2 would have implied that we can describe, in a global way, a universal routing scheme providing a routing function $R$ for any graph $G$ of $n$ vertices, with a total memory requirement of $\Theta(n^2)$ bits. Indeed, we can construct $R = (I, H, P)$ as follows: let $M$ be the adjacency matrix of a given graph $G = (V, E)$. The initialization function is $I = \{I_x = Identity \mid x \in V\}$, the header function is $H = \{H_x = Identity \mid x \in V\}$, and the header set $\mathcal{H} = V$. The port function $P$ returns for any pair of vertices $(x, y)$ the arc $e = (x, z) \in E$ such that $e$ belongs to a shortest path from $x$ to $y$. The function $P$ is only an algorithm for finding the shortest path that uses the matrix $M$. Moreover $R$ is of stretch factor 1 and the size of all the headers is bounded by $\lceil \log_2 n \rceil$ bits. Unfortunately Theorem 2 does not give the way to distribute the $\Theta(n^2)$ bits of information needed for such a routing function $R$ among the vertices.

### 4.5 Proof of Theorem 1

Let $n$ be any integer, $n \geq 20$, let $s$ be any real, $1 \leq s < 2$, and let $RS^*$ be a "best" universal routing scheme, *i.e.* a routing scheme such that for every graph $G$ of order $n$, denoted by $\mathcal{G}$, $\text{MEM}_{global}(G, s) = \text{MEM}_{global}(G, s, RS^*(G))$. Applying Theorem 2, we get $|RS^*(\mathcal{G})| \in 2^{\Theta(n^2)}$. Therefore, for every $n \geq 20$, there exists a graph $G$ of order $n$, such that $\text{MEM}_{global}(G, s) \in \Omega(n^2)$, because $RS^*(G)$ is the routing function that requires the minimum memory for the graph $G$. □

## 5 Conclusion

In this paper, we proved that for any stretch factor $s$, $1 \leq s < 2$, $\text{MEM}_{global}(n, s) \in \Omega(n^2)$ and $\text{MEM}_{local}(n, s) \in \Omega(n)$. It means that any near shortest path routing scheme uses a total of $\Omega(n^2)$ memory bits, and locally uses at least $\Omega(n)$ memory bits on a router. On the other hand, we know that $\text{MEM}_{global}(n, 1) \in O(n^2 \log n)$ and $\text{MEM}_{local}(n, 1) \in O(n \log n)$ by using tables. An interesting question would be to fill the gap between these bounds: is it possible to be globally and/or locally more compact than the tables ?

Now, though this last question is of high theoretical interest, the bounds only differ in a factor of $\log n$. A more practical question would be to identify for any given constant $k$ the class of networks $G$ such that the memory requirement of $G$ is less than $k$. The same question is worth answering for $k$ being replaced by a slowly increasing function (such as $\log n$). The reverse question is also worth answering: given a class of networks, what is the maximum memory requirement for this class. For instance, we know that $\max\{\text{MEM}_{local}(G, 1), G \text{ graph of order } n \text{ and maximum degree } \Delta\} \in O(n \log \Delta)$, but we do not know yet lower bounds for this class of networks. We think that it should be possible to find an efficient shortest path routing scheme for this particular class of networks (that uses even less than $O(n)$ memory bits locally).

## References


[1] E. M. Bakker, J. van Leeuwen, and R. B. Tan, *Linear interval routing*, Algorithms Review, 2 (1991), pp. 45–61.

[2] W. J. Dally and C. L. Seitz, *The torus routing chip*, Distributed Computing, 1 (1986), pp. 187–196.

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

[4] F. Desprez, E. Fleury, and M. Loi, *T9000 et C104 : La nouvelle génération de transputers*, Research Report 93-01, LIP, École Normale Supérieure de Lyon, 69364 Lyon Cedex 07, France, Feb. 1993.

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

[6] P. Fraigniaud and C. Gavoille, *Interval routing schemes*, Research Report 94-04, LIP, École Normale Supérieure de Lyon, 69364 Lyon Cedex 07, France, Jan. 1994.

[7] ———, *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.

[8] P. Fraigniaud and E. Lazard, *Methods and problems of communication in usual networks*, Discrete Applied Maths, 53 (1994), pp. 79–133.

[9] P. Fraigniaud, A. Liestman, and D. Sotteau, *Open problems*, Parallel Processing Letters, 3 (1993), pp. 507–524.

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

[11] ———, *Efficient message routing in planar networks*, SIAM Journal on Computing, 18 (1989), pp. 843–857.

[12] ———, *Space-efficient message routing in c-decomposable networks*, SIAM Journal on Computing, 19 (1990), pp. 164–181.

[13] 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.

[14] M. Gondran and M. Minoux, *Graphs and Algorithms*, John Wiley, 1984.

[15] M. A. Harrison, *On the number of classes of binary matrices*, IEEE Transactions on Computers, C-22 (1973), pp. 1048–1051.

[16] M. Heydemann, J. Meyer, and D. Sotteau, *On forwarding indices of networks*, Discrete Applied Mathematics, 23 (1989), pp. 103–123.

[17] E. Kranakis, D. Krizanc, and S. S. Ravi, *On multi-label linear interval routing schemes*, in $19^{th}$ International Workshop on Graph - Theoretic Concepts in Computer Science - Distributed Algorithms (WG), vol. 790 of Lecture Notes in Computer Science, Springer-Verlag, June 1993, pp. 338–349.

[18] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes*, Morgan Kaufmann, 1992.

[19] D. Peleg and E. Upfal, *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.

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

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

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