Memory Requirement for Routing in Distributed Networks*
(Extended Abstract)

Cyril Gavoille†
LIP, École Normale Supérieure de Lyon
(gavoille@ens-lyon.fr)

Stéphane Pérennès‡
I3S, Université de Nice-Sophia Antipolis
(sp@essi.fr)

Abstract

In this paper, we deal with the compact routing problem on distributed networks, that is implementing routing schemes that use a minimum memory size on each node. We prove that for every shortest path routing scheme, for any constant ε, $0 < \varepsilon < 1$, and for every integer d such that $3 \leq d \leq \varepsilon n$, there exists an n-node network of maximum degree d that locally requires $\Theta(n \log d)$ bits of memory on $\Theta(n)$ nodes. This optimal lower bound means that whatever you choose the routing scheme (interval routing, boolean routing, prefix routing, ...), there exists a network on which one can not do better than routing tables.

1 Introduction

1.1 The Compact Routing Problem

The *general routing problem* in a network consists of finding a routing *protocol*, or routing *function*, such that, for any pair source-destination, any message from the source can be routed to the destination. In the *compact routing problem* we are interested to implement protocols that require a low amount of hardware and easily to obtain in a VLSI point of view.

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 necessary have the simple structure of meshes or hypercubes. 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, and can be applied on all networks. However each router needs to store a large table of size $O(n \log d)$ bits for an n-node network of maximum degree d. Moreover, since routers must be replicated n times in the network, such a large local memory requirement implies a global information of $O(n^2 \log d)$ bits for the whole network, just for routing.

Many authors proposed methods to compress the routing table (see for instance [11, 12, 19]). Among them, *interval routing* [3, 6, 19, 20] was chosen by *Inmos* for its C104 routing chip [5]. Unfortunately, it was proved in [6, 17] 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 proved in [14] that this number is $\Theta(n)$ for an unbounded degree n-node network). 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 (or a routing function) 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, as routing tables do. The shortest path interval routing scheme is therefore called *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 require routers to implement the routing functions of this scheme. Partial routing schemes require less memory, but they do not apply on all networks. In the last PODC Symposium, in [8], it is proved that any universal routing scheme on n-node networks, requires a total of $\Omega(n^2)$ bits of memory to store a shortest path routing function on some networks. In fact, the worst-cases networks that require the total of $\Omega(n^2)$ bits of memory have a unbounded maximum degree, namely $\Theta(n)$.

*Part of this work has been done while the authors were visiting the Weizmann Institute of Israel. Additional support by the French-Israel association AFIRST.

†Supported by the research programs PRS and ANM of the CNRS, and by the DRET.

‡Supported by the research program PRS of the CNRS.

In this paper the general problem of the memory requirement for shortest path routing scheme on n-node networks of maximum degree d is addressed. We prove that, for any constant ε, $0 < \varepsilon < 1$, and for every integer d, $3 \leq d \leq \varepsilon n$, there exists an n-node network of maximum degree d on which any shortest path routing function requires $\Theta(n \log d)$ bits of local memory on $\Theta(n)$ nodes. This proves that, in general, one can not do better than routing tables for $\Theta(n)$ nodes. It means that whatever the way you choose the shortest paths on this network, the labels of the nodes in the space $\{1, \ldots, n\}$ and whatever the way you choose the port labeling, the total routing information is $\Omega(n^2 \log d)$ bits. In other words, there is no hope to find a universal routing scheme more compact than routing tables for n-node networks of maximum degree d. This general result improves previous works on the compact routing problem studied in [8, 16, 18].

1.2 A General Model of Routing Function

A point-to-point communication network is described by a finite 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 nodes or the routers of the network: we assume that one router is assigned to each processor (PE). The model may be easily extended to many processors connected to a router. The edges represent bidirectional communication links between the routers (to each edge corresponds two opposite arcs). A special bidirectional link connects the processor to the router (see Figure 1). A vertex can communicate directly only with its neighbors, and messages between nonadjacent vertices are sent along a path connecting them in the network. The router is composed of an input and of an output *interface* and of a *local routing function*. The input interface sends to the local routing function the input port p and the header h of an arriving message M entering the router. The local routing function computes a new header h' and a new output port p', and sends these informations to the output interface. Finally, the output interface forwards M and its new header h' on the port p'. This model can be used to describe an universal routing scheme.

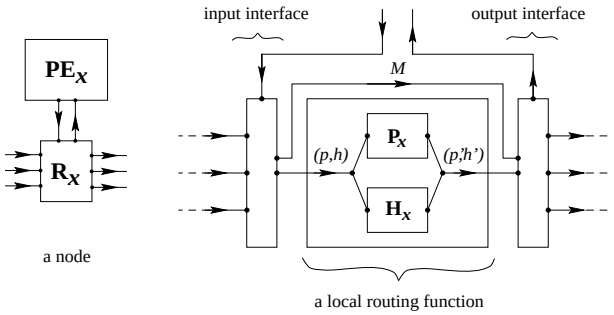

Figure 1: A model of router

In a sense, finding a routing function in a symmetric digraph $G = (V, E)$ consists on label vertices (= routers) and arcs (= ports of routers) of G such that the local routing function can compute a path between any pair source-destination of vertices (our goal being to measure the minimum number of bits to code and to store the local routing function in a router). More precisely a *vertex-labeling* of G is an injection $\mathcal{L} : V \to I\!N$, that assigns for every vertex $x \in V$ an unique integer $\mathcal{L}(x) \in I\!N$. Let us denote by E_x the set of arcs incident to x for every $x \in V$. A *local arc-labeling* of a vertex x of G is a function $\mathcal{E}_x : E_x \to I\!N$, such that any two arcs incident to x have the same label if and only if they represent the same edge (= bidirectional link). It means that we label by the same port an input link and output link of a same edge incident to x. The collection $\mathcal{E} = \{\mathcal{E}_x \,|\, x \in V\}$ is called an *arc-labeling* or a *port labeling* of G. Without loss of generality, we assume that headers are coded by positive integers. Thus, the sets of labels (vertices and ports) and headers are subsets of $I\!N$. Without loss of generality, we also assume that links between the router and the PE are labeled 0. Definition 1 gives a formal definition of a routing function on a graph G. Note that it is slightly different from the model given by Peleg and Upfal in [18] (the initialization function is not considered).

Definition 1 (Routing Function) *Let $G = (V, E)$ be a symmetric digraph. A* routing function *on G is a function $R : V \times I\!N \times I\!N \to I\!N \times I\!N$ such that there exists a vertex-labeling $\mathcal{L}$ and an arc-labeling $\mathcal{E} = \{\mathcal{E}_x \,|\, x \in V\}$ satisfying: for every couple (x, y) of vertices of G, there exist:*

- *a sequence $\{v_i \in V \,|\, 0 \leq i \leq k\}$ of $k+1$ vertices, forming a path from x to y in G*
- *a sequence $\{p_i \in I\!N \,|\, 0 \leq i \leq k\}$ of $k+1$ port labels, labeling the arcs between two consecutive vertices of the path*
- *a sequence $\{h_i \in I\!N \,|\, 0 \leq i \leq k\}$ of $k+1$ headers, obtained by successive modification of the original header along the path*

such that:

- $v_0 = x$, $p_0 = 0$, $h_0 = \mathcal{L}(y)$
- $(p_i, h_i) = R(v_{i-1}, p_{i-1}, h_{i-1})$, *for every* $1 \leq i < k$
- $v_k = v_{k-1} = y$, $p_k = 0$, $h_k = h_{k-1}$

Remark. In this paper we focus on finite connected symmetric digraph without multi-arc. In the following we always refer to such digraphs by the simple term *graph*.

From Definition 1, if R is a routing function on a graph $G = (V, E)$, the path is built by R as follows. If the processor assigned to the router x wants to send a message M to the processor assigned to the router y, it sends the label of the destination, $\mathcal{L}(y)$, by its special link labeled 0. The router x prepares a header h and an output port p by computing the pair $(p, h) = R(x, 0, \mathcal{L}(y))$. The header h is attached to the message M, and both are sent on port p. Now, suppose that the message M with a header h' arrives at some router z by the port p'. The router computes $(p'', h'') = R(z, p', h')$. If $p'' = 0$, the routing process ends, *i.e.* the message M with the header $h' = h''$ is sent on the special link 0 to the processor assigned to the router z. Otherwise, z must forward the message M with the new header h'' (replacing the old header h'), attached to the message, and sends the message on port p''.

Notations. The distributed representation of R on $G = (V, E)$ is given by the collection $\{R_x \,|\, x \in V\}$ such that

$R_x(p,h) = R(x,p,h)$, for every $x \in V$, and every $(p,h) \in \mathbb{N}^2$. For every $x \in V$, R_x is called the *local routing function* of R in x. Moreover R can be viewed as a pair of functions (P,H), where P is the *port* function and where H is the *header* function. Similarly P_x and H_x denote the *local* port function and the *local* header function respectively (see Figure 1). They are defined by $P_x(p,h) = P(x,p,h)$ and $H_x(p,h) = H(x,p,h)$, for every $x \in V$ and every $(p,h) \in \mathbb{N}^2$.

In this model, we consider oblivious routing functions, *i.e.* routing functions R such that the path induced by R from x to y depends on x and y only. Such functions are also called *static* or *non-adaptive* routing functions. Of course the results of the local header function H_x used to compute the next header depend on the current header and the entering port of the message. Similarly, the results of a local port function P_x also depend on this header and this port. This allows to employ flexible *trial-and-error* routing functions, that is the paths induced by R can contain loops. This property may be able to decrease the local memory requirement for the storage of routing information because it allows to visit many nodes to gather informations that are stored in the headers. This possibility has been efficiently used in the approach of [18]. Since our model is more general, our lower bounds on the memory requirement also apply for the usual routing functions where headers always contain the name of the destination only.

The length of the path from x to y induced by R is the number of arcs of G used by the path. A routing function R is said a shortest path routing function if and only if the paths induced by R are always shortest paths in G. In this paper, we consider shortest path routing function only. Tradeoffs between memory requirement and length of the routing path have been studied in [1, 2, 18]. However, lower bounds given in these papers are not tight for near-shortest path routing schemes, and do not include the maximum degree of the network as parameter.

1.3 Memory Requirement

Definition 2 (Memory Requirement of a Vertex) *Given a graph G, a routing function R on G, and a vertex x of G, the* memory requirement *of x, denoted by* $\mathrm{MEM}(G,R,x)$, *is the minimum number of bits necessary to store the local routing function R_x.*

In a sense the memory requirement of a vertex x is the information needed to evaluate R_x on some inputs, or the minimum ROM (Read Only Memory) size necessary to describe R_x. It can be formally defined as the Kolmogorov complexity of the function R_x defined as the size of the smallest program that returns R_x and halts [15]. Just as the Kolmogorov complexity, the memory requirement is defined for a fixed programming language. Of course computation of the function R_x itself may require RAM (Random-Access Memory) that is not taken in account here. The *memory requirement* is only a lower bound on the space memory required by a router.

Clearly, using some tricks, every graph G and every vertex x of G supports a routing function R_x that can be described with at most $O(1)$ bits. Indeed labelings (vertex and port) are not taken into account in the memory requirement. Therefore we could use the label of each vertex to encode the full routing information (including for example the adjacency matrix of G) and each port label to store the label of a node. This approach reduces memory size of the router to a constant, but forces the *PE*s to know what are vertex-labels available in the network, that may waste a lot of bits in the inner *PE*'s memory. This is why we will limit in this paper the class of routing functions to those that make use of a vertex-labeling from $\{1,\ldots,n\}$ and a local arc-labeling from $\{1,\ldots,\deg(x)\}$, where $\deg(x)$ is the degree of x. Note that the limit to $\{1,\ldots,n\}$ for the vertex-labeling corresponds to the model of many compact routing schemes, including the interval routing schemes, and is implicitly used in the literature for lower bounds (*e.g.* see [18] and [8]). However, certain upper bounds use longer labels (see [1, 2, 4, 11, 12, 18]).

Under this assumption, let us consider the complete graph K_n of order n. In general, in a vertex x, a local routing function R_x on K_n can be stored with $\Theta(n \log n)$ bits for a random port labeling of x or for 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 a such way that to reach 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, some routing function R'_x on K_n may be stored in x in a more compact way using $O(\log n)$ bits. For example vertices are labeled by integers from $\{1,\ldots,n\}$ and ports by integers from $\{1,\ldots,n-1\}$ such that the ports connecting vertices x and y is the integer y if $x > y$ and $y-1$ if $y > x$. Note that we do not make difference between the vertex x and its label. If headers are always labels of destination, the routing function in x, $R'_x = (P'_x, Identity)$, is defined by its port function P'_x. It runs the following algorithm: if the destination is x then the message is arrived, otherwise forward the message on the port that has the same name as the destination if $x > y$, else on the port $y-1$. Clearly $R' = \{R'_x = (P'_x, H'_x) \mid x \in V\}$ defines a shortest path routing function on K_n, and the previous algorithm can be described with only $O(\log n)$ bits for the item x. In general we are interesting to find the "best" routing function on a given graph, that is the best vertex and port labeling, and the best choice of routing paths that minimize coding of the routing function. These examples lead to the following definition of the memory requirement of a graph.

Definition 3 (Memory Requirement of a Graph) *Given a graph $G = (V,E)$, the* memory requirement of the graph G *is the value denoted by* $\mathrm{MEM}_{local}(G)$ *:*

$$\mathrm{MEM}_{local}(G) = \min_{R} \max_{x \in V} \mathrm{MEM}(G,R,x)$$

where R is minimized over all shortest path routing functions on G, such that vertex-labels are from $\{1,\ldots,|V|\}$ and port labels of every $x \in V$ are from $\{1,\ldots,\deg(x)\}$.

The compact routing problem consists of finding a routing scheme that, for every graph G, returns a routing function on G that locally needs the minimum number of bits. For example the memory requirement for H_n, the hypercube of order n, is $O(\log n)$, by using e-cube routing. It is known that all trees [19], outerplanar graphs [10] and all unit circular-arc graphs [7] of order n and of maximum degree d,

have a memory requirement of $O(d \log n)$ bits, because the 1-interval routing scheme applies to these graphs (1 interval per arc). Note that our approach is *local*.

To prove lower bounds for the memory requirement, we made the model as general as possible. We summarize below some consequences of Definitions 1, 2 and 3:

(1) Headers are of unrestricted size.
(2) Routing functions are of shortest paths.
(3) Networks are finite connected symmetric digraphs without multi-arc and with unit[1] cost on the arcs.
(4) Vertex are labeled by integer in $\{1, \ldots, n\}$.
(5) For every vertex x, ports of x are labeled by integer in $\{1, \ldots, \deg(x)\}$.

1.4 Lower Bound on the Memory Requirement

This section shortly presents the main results of this paper.

Theorem 1 *Let ε, $0 < \varepsilon < 1$, be a fixed constant. For every integer d, $3 \leq d \leq \varepsilon n$, there exists a graph of order n and maximum degree d that has a memory requirement of $\Theta(n \log d)$ bits on $\Theta(\log_d n)$ vertices.*

To prove this result we construct a graph where a small part of the vertices, namely $\Theta(\log_d n)$, needs to know all together a large amount of routing information, namely a total of $\Omega(n \log n)$ bits, to route along shortest paths. This graph of order n and maximum degree d is denoted by $G_{n,d}$. The proof is given in Section 3.

In Section 4 we prove a general result that allows to increase the number of routers having to know $\Omega(n \log d)$ bits of routing information. This transformation, applied to a $G_{\Theta(n),d}$ graph allows to derive a graph of order n and maximum degree d that is denoted by Split-$G_{n,d}$. The following corollary is proved in Section 4.

Corollary 1 *Let ε, $0 < \varepsilon < 1$, be a fixed constant. For every integer d, $3 \leq d \leq \varepsilon n$, there exists a graph of order n and maximum degree d that has a memory requirement of $\Theta(n \log d)$ bits on $\Theta(n)$ vertices.*

This corollary proves that there exists a case where the global routing information is $\Theta(n^2 \log d)$ bits, implying that routing tables are *incompressible* in the general case, or that there is no *compact* distributed routing algorithm for this network.

We present below a table that summarizes bounds on the memory requirement, and the number of routers using such a requirement as a function of the maximum degree d and the order of the graph. The second table is the sum of the memory requirement on all the routers of these graphs. It is well known that every graph of maximum degree 2 (paths P_n and rings C_n) has $O(\log n)$ bits of memory requirement on all vertices, because a 1-interval routing scheme can be applied (see [20]). In this table, ε is an arbitrary positive constant < 1.

[1] Of course our lower bounds can be applied to the memory requirement of networks with non uniform costs on its links.

maximum degree	memory bits	# routers	graphs
2	$O(\log n)$	n	P_n, C_n
$3 \leq d \leq \varepsilon n$	$\boldsymbol{\Theta(n \log d)}$	$\Theta(\log_d n)$	$G_{n,d}$
$3 \leq d \leq \varepsilon n$	$\boldsymbol{\Theta(n \log d)}$	$\boldsymbol{\Theta(n)}$	Split-$G_{n,d}$
$n-1$	$O(\log n)$	n	K_n

graphs	total memory bits
P_n, C_n	$O(n \log n)$
$G_{n,d}$	$\Omega(n \log n)$
Split-$G_{n,d}$	$\boldsymbol{\Theta(n^2 \log d)}$
K_n	$O(n \log n)$

2 Matrices and Graphs of Constraints

Complexity of proofs is due to the fact that for each graph we allow to choose the set of shortest paths, the vertex-labeling and the port labeling that minimize the local routing information.

The main idea of our lower bounds is based on the fact that in some graph G some vertices can be connected by a unique shortest path. Communications between these vertices are *constrained*, *i.e.* every shortest path routing function on G has to select the same arc for communication between such a pair of vertices. A set of constraints is an invariant information for routing on G, under any vertex and port labelings. Then we try to build graphs from sets of constraints. More precisely, for each set of constraints, called *matrix of constraints*, we build a graph that possesses a set of unique shortest path satisfying these constraints. We argue that certain set of constraints (even after any vertex and port permutation) needs a large memory to be encoded in comparison with the size of the graph built. Roughly speaking, every routing function on such a graph requires to store this amount of memory.

Recall the notion of matrix and graph of constraints introduced in [9]. The basic idea is to apply a counting argument to a set of graphs on which any shortest path routing function "computes" a finite set of integer matrices.

Definition 4 (Matrix of Constraints) *Let $G = (V, E)$ be a graph. A* matrix of constraints *of G is a $p \times q$ integer matrix $M = (m_{i,j})$, $i \in I = \{1, \ldots, p\}$ and $j \in J = \{1, \ldots, q\}$, such that there exist two sets of vertices $A = \{a_i \mid i \in I\} \subset V$ and $B = \{b_j \mid j \in J\} \subset V$, satisfying that, for every shortest path routing function $R = \{(P_x, H_x) \mid x \in V\}$ on G, there is a way to label the incident arcs of vertices of A such that:*

$$\forall (i,j) \in I \times J, \quad m_{i,j} = P_{a_i}(0, b_j)$$

Informally if $M = (m_{i,j})$ is a matrix of constraints of a graph G, Definition 4 means that there exists two subsets of vertices, A and B, such that every shortest path from $a_i \in A$ to $b_j \in B$ in G starts with the arc $e_{i,j}$ which can be labeled by the port $m_{i,j}$. In the following vertices of A are called *constrained vertices* and vertices in B are called *target vertices* for M.

Definition 5 ($\sim$ Relation) *Let $M = (m_{i,j})$ and $M' = (m'_{i,j})$ be two $p \times q$ integer matrices, $i \in I = \{1, \ldots, p\}$ and $j \in J = \{1, \ldots, q\}$. $M \sim M'$ if and only if there exist two permutations r of I and c of J, and p permutations ψ_i of $\{1, \ldots, |\cup_{j \in J} \{m_{i,j}\}|\}$, $i \in I$, such that:*

$$\forall (i,j) \in I \times J, \quad m'_{i,j} = \psi_i(m_{r(i),c(j)})$$

The relation $\sim$ is an equivalence relation since r, c and ψ_i, for $i \in \{1, \ldots, p\}$, are bijective functions. $M' \sim M$ means that M' is obtained from M by row and column permutation, and by a permutation of the set of entries for each row of M. Permutations ψ_i can be seen as a port labeling whereas r and c can be seen as a vertex-labeling (precisely, vertex-labeling of constrained and target vertices).

Notation. For every set $\mathcal{M}$ of $p \times q$ integer matrices, we denote by $\mathcal{M}_{/\sim}$ the set of equivalence classes of $\mathcal{M}$ under the equivalence relation $\sim$.

Definition 6 (Graphs of Constraints) *Let $\mathcal{M}$ be a set of $p \times q$ integer matrices. A set of graphs $\mathcal{G}$ is a set of* graphs of constraints *of $\mathcal{M}$ if it satisfies that for every $M \in \mathcal{M}$ there exists a graph $G \in \mathcal{G}$ that possesses M as matrix of constraints.*

The following lemma establishes the fundamental property of graphs of constraints relative to the memory requirement. It proves that graphs of constraints of $\mathcal{M}$, if exist, must require some information depending of $|\mathcal{M}_{/\sim}|$ and the number of constrained vertices. Theorem 1 is based on this lemma.

Lemma 1 *Let $\mathcal{M}$ be a set of $p \times q$ integer matrices. For any set $\mathcal{G}$ of graphs of constraints of $\mathcal{M}$, there exists a graph $G \in \mathcal{G}$ such that the sum of the memory requirement of p vertices of G is at least*

$$\log_2 \frac{|\mathcal{M}_{/\sim}|}{\binom{n}{q}} - O(\log n)$$

where n is the order of G.

Sketch of the proof. Let $M \in \mathcal{M}$, there exists $G = (V, E) \in \mathcal{G}$ having n vertices, and two subsets of vertices A, B of size p and q respectively, such that for every routing function on G, A consists of the constrained vertices and B consists of the target vertices for M. Let $R = \{R_x = (P_x, H_x) \,|\, x \in V\}$ be any shortest path routing function on G, let $R_A = \{R_a = (P_a, H_a) \,|\, a \in A\}$ and let $f_B : \{1, \ldots, n\} \to \{0, 1\}$ the characteristic sequence of the set B in $\{1, \ldots, n\}$ ($f_B(x) = 1$ if and only if there exists a vertex $b \in B$ that is labeled x). We show that the knowledge of R_A, f_B and $O(\log n)$ bits is sufficient to compute a matrix $M' \sim M$. Indeed, we have no information about the vertex and the port labeling of G (in particular of vertices of A) minimizing the coding of functions of R_A. Let m the minimum number of bits to necessary store R_A. We show that $m + \log_2 \binom{n}{q} + O(\log n) \geq \log_2 |\mathcal{M}_{/\sim}|$, and therefore that the sum of the memory requirement on p routers is at least m bits where $m \geq \log_2(|\mathcal{M}_{/\sim}|/\binom{n}{q}) - O(\log n)$. □

Remark. If all graphs of $\mathcal{G}$ are of order at most n then one can precise Lemma 1, based on a simple counting argument, showing that at least $|\mathcal{M}_{/\sim}|/2$ graphs of $\mathcal{G}$ have a sum of the memory requirement on p vertices of at least $\log_2(|\mathcal{M}_{/\sim}|/\binom{n}{q}) - O(\log n)$ bits.

3 Graph Constructions

In this section we will construct a set of graphs of constraints of a particular set of matrices with maximum cardinality in order to apply Lemma 1. First to sharpen the proof of Theorem 1, we give a basic construction for low degree $3 \leq d \leq n^{1/3}$ and whenever $n = \alpha(d-1)^p$, for any constant $\alpha \geq 10$ and for any integer $p \geq 2$. Then we generalize the construction for every n and for every degree d.

3.1 Construction for n power of $\delta = (d-1)$

For every pair of integers (p, δ), let $A_{p,\delta} = (a_{i,j})$ be the integer matrix such that $a_{i,j} = (\lfloor (j-1)/\delta^{p-i+1} \rfloor \bmod \delta) + 1$, for every $i \in \{1, \ldots, p\}$ and for every $j \in \{1, \ldots, \delta^p\}$. Each column of $A_{p,\delta}$ can be seen as a word of length p from the alphabet $\{1, \ldots, \delta\}$, and the set of columns can be seen as the set of all different words of length p from the alphabet $\{1, \ldots, \delta\}$. For example

$$A_{2,3} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 \end{bmatrix}$$

Lemma 2 *For every pair of integers (p, δ), $p \geq 1$ and $\delta \geq 2$, there exists a graph $H_{p,\delta}$ that satisfies:*

(1) *$H_{p,\delta}$ has at most $5\delta^p$ vertices*
(2) *$H_{p,\delta}$ is of maximum degree $\delta + 1$*
(3) *$H_{p,\delta}$ possesses $A_{p,\delta}$ as matrix of constraints*

Sketch of the proof. The graph $H_{p,\delta}$ is defined by induction on p. $H_{p,\delta}$ has two distinguished subsets of vertices: A_p, the set of constrained vertices, that are labeled $\{1, \ldots, p\}$, and B_p, the set of target vertices, that are labeled by all the words of length p from the alphabet $\{1, \ldots, \delta\}$. $H_{1,\delta}$ is isomorph to $K_{1,\delta}$. We denote by $T_{p,\delta}$, the complete δ-ary tree of height p and of δ^p leaves. $H_{p+1,\delta}$ is composed of a copy of $H_{p,\delta}$ and a copy of $T_{p,\delta}$ connected by $\delta^p + \delta^{p+1}$ intermediate vertices (see Figure 2). For the particular case $p + 1 = 2$, we do not add the first $\delta^p = \delta$ intermediate vertices. Each vertex of the first δ^p of intermediate vertices is connected by an edge to a vertex of B_p, forming a new set of vertices denoted B'_p labeled like B_p. The δ^{p+1} other vertices form the set B_{p+1} of target vertices of $H_{p+1,\delta}$ labeled by all the words of length $p + 1$ from the alphabet $\{1, \ldots, \delta\}$. The δ^p leaves of $T_{p,\delta}$ are partitioned in δ subsets, each one of p leaves. A leaf x is labeled i, for every $i \in \{1, \ldots, \delta\}$, if the i^{th} child of the root of $T_{p,\delta}$ is an ancestor of x. Each vertex $w = ui$ of B_{p+1}, where i is a word of length p and x is a letter of the alphabet $\{1, \ldots, \delta\}$, is of degree two and is connected to $u \in B'_p$ and to one of the leaf x labeled i, such that x is connected to at most δ vertices of the set B_{p+1}. □

Figure 3 shows an example of the $H_{p,\delta}$ graph.

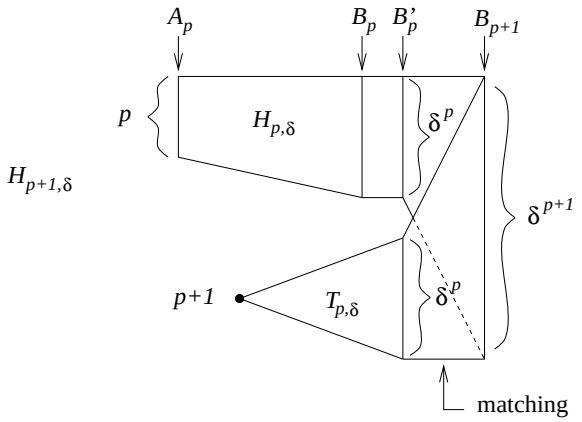

Figure 2: The recursive construction of the $H_{p,\delta}$ graph.

Lemma 3 *For every pair of integers (p,δ), let $\mathcal{A}_{p,\delta}$ be the set of $2p \times \delta^p$ integers matrices of the form $\begin{bmatrix} A_{p,\delta} \\ B \end{bmatrix}$ where $B \sim A_{p,\delta}$. Then, for every $p \geq 2$ and for every $\delta \geq 2$,*

$$\frac{(\delta^p)!}{(\delta!)^{2p}(2p)!} \leq |\mathcal{A}_{p,\delta/\sim}| = 2^{\Omega(\delta^p \log \delta^p)}$$

Proof. Let (p,δ) a pair of integers ≥ 2 and let $\mathcal{A} = \{M \mid M \sim X \in \mathcal{A}_{p,\delta}\}$. Then $|\mathcal{A}| \geq (\delta^p!)^2$, because any matrix of the form $M = \begin{bmatrix} \pi_1(A_{p,\delta}) \\ \pi_2(A_{p,\delta}) \end{bmatrix}$, where π_1, π_2 are two permutations of columns of each submatrix $A_{p,\delta}$, is in $\mathcal{A}$. Indeed, apply π_1^{-1} on the column of M to get $\begin{bmatrix} A_{p,\delta} \\ \pi_1^{-1}(\pi_2(A_{p,\delta})) \end{bmatrix}$. Therefore $|\mathcal{A}_{/\sim}| \geq (\delta^p)!^2/((\delta^p)!(\delta!)^{2p}(2p)!)$, where $(\delta^p)!(\delta!)^{2p}(2p)!$ is the maximum size of a set of $2p \times \delta^p$ integer matrices from $\{1,\ldots,\delta\}$ equivalent under $\sim$ (clearly $\mathcal{A}$ is a set of integer matrices from $\{1,\ldots,\delta\}$). Moreover $|\mathcal{A}_{/\sim}| = |\mathcal{A}_{p,\delta/\sim}|$, thus $|\mathcal{A}_{p,\delta/\sim}| \geq (\delta^p)!/((\delta!)^{2p}(2p)!)$. Let $L = \log_2 |\mathcal{A}_{p,\delta/\sim}|$. We get $L = \Omega(p\delta^p \log\delta - p\delta\log\delta - p\log p) = \Omega(p((\delta^p - \delta)\log\delta - \log p))$. Since $\delta \geq 2$ and $p \geq 2$, $L = \Omega(p\delta^p \log\delta) = \Omega(\delta^p \log \delta^p)$. □

The following proposition shows that some routers of a graph, called the $G_{n,d}$ graph (see figure 4), can require $\Omega(n \log d)$ bits.

Proposition 1 *Let α be a real constant, $\alpha \geq 10$, and p be a constant integer, $p \geq 2$. For every integer d, $3 \leq d \leq n^{1/3}$, there exists a graph of order $n = \alpha(d-1)^p$ and maximum degree d that has a memory requirement of $\Theta(n \log d)$ bits on $\Theta(\log_d n)$ vertices.*

Sketch of the proof. We build a set of graphs of constraints of order n and maximum degree d of the set of matrices $\mathcal{A}_{p,d-1}$ (introduced in Lemma 3) with $2p$ constrained vertices. The graph is composed of two copies of $H_{p,\delta}$ connected by a perfect matching between target vertices of the $H_{p,\delta}$ graphs (see Figure 4). The target vertices

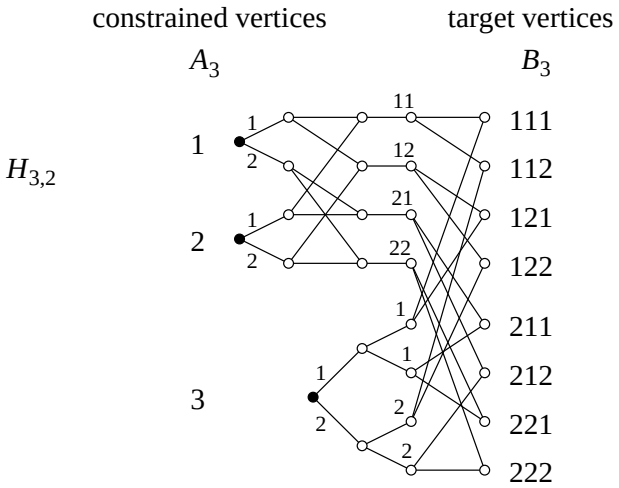

Figure 3: An example of the $H_{p,\delta}$ graph for $p = 3$ and $\delta = 2$. The shortest paths leading to the target vertex $xyz \in B_3$ have to use the port x from the constrained vertex $1 \in A_3$, the port y from the vertex 2 and the port z from the vertex 3.

of the whole graph are target vertices of only one copy of $H_{p,\delta}$. Roughly speaking, applying Lemma 1 and Lemma 3, the sum of the memory requirement on these $2p$ vertices is $\Omega((d-1)^p \log (d-1)^p) - O(n)$. The term $O(n)$ is because $\log_2 \binom{n}{x} \leq n$ for every x. Since $n = \alpha(d-1)^p$ for a constant α, it follows that $n = \Theta((d-1)^p)$. Therefore the sum of the memory requirement is $\Omega(n \log n)$ on $2p = \Theta(\log_{d-1} n) = \Theta(\log_d n)$ vertices, for $d \geq 3$. □

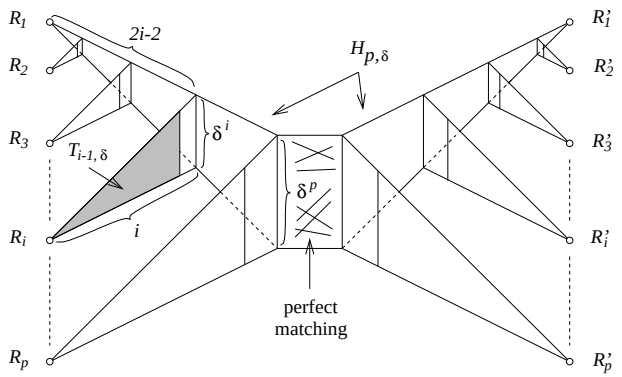

Figure 4: The $G_{n,d}$ graph. A graph of constraints of order n and of maximum degree $d = \delta + 1$. Routers $R_1,\ldots,R_p$ and $R'_1,\ldots,R'_p$ "compute" the perfect matching between $2\delta^p = \Theta(n)$ vertices.

3.2 Construction for every n and for every d

We generalize the $A_{p,\delta}$ matrices by the $A_{p,\delta,k}$ matrices in the following way: $A_{p,\delta,k}$ is a $(p+1) \times (k\delta^p)$ integer matrix

composed of k $A_{p,\delta}$ matrices, $[A_{p,\delta}\, A_{p,\delta}\, \ldots\, A_{p,\delta}]$, plus a row $[1\,1\,\ldots\,1\,2\,2\,\ldots\,2\,\ldots\,k\,k\,\ldots\,k]$. Each column of the $A_{p,\delta,k}$ matrix can be seen as a word of length $p+1$, where the first symbol is from the alphabet $\{1,\ldots,k\}$ and the p last are from the alphabet $\{1,\ldots,\delta\}$. Note that the $A_{p,\delta}$ matrices are particular cases of the $A_{p,\delta,k}$ matrices since $A_{p,\delta,\delta} = A_{p+1,\delta}$. For example

$$A_{2,3,2} = \left[\begin{array}{ccccccccc|ccccccccc} 1&1&1&1&1&1&1&1&1&2&2&2&2&2&2&2&2&2\\ 1&1&1&2&2&2&3&3&3&1&1&1&2&2&2&3&3&3\\ 1&2&3&1&2&3&1&2&3&1&2&3&1&2&3&1&2&3 \end{array}\right]$$

Lemma 4 *For every triple of integers (p,δ,k), $p \geq 1$ and $\delta \geq k \geq 2$, there exists a graph $H_{p,\delta,k}$ that satisfies:*

(1) *$H_{p,\delta,k}$ has at most $(k+8)\delta^p$ vertices*
(2) *$H_{p,\delta,k}$ is of maximum degree $\delta+1$*
(3) *$H_{p,\delta,k}$ possesses $A_{p,\delta,k}$ as matrix of constraints*

Sketch of the proof. The $H_{p,\delta,k}$ graph is similar to a $H_{p+1,\delta}$ graph where the tree $T_{p,\delta}$, that was mixed to $H_{p+1,\delta}$ by a matching, is replaced by a new tree denoted by $Q_{p,\delta,k}$ where its root is of degree k. More precisely $H_{p,\delta,k}$ is composed of $H_{p,\delta}$ with δ^p edges added to the target vertices of $H_{p,\delta}$, a tree $Q_{p,\delta,k}$ and $k\delta^p$ additional vertices (that are the target vertices of $H_{p,\delta,k}$) that are connected by a matching. The tree $Q_{p,\delta,k}$ is composed of k δ-ary subtrees, each having δ^{p-1} leaves at equal distance from its root. These k subtrees are connected to a single vertex of degree k, that forms the root of $Q_{p,\delta,k}$. Figure 5 shows a $H_{2,3,2}$ graph that possesses the matrix $A_{2,3,2}$ (put on page 7) as matrix of constraints. □

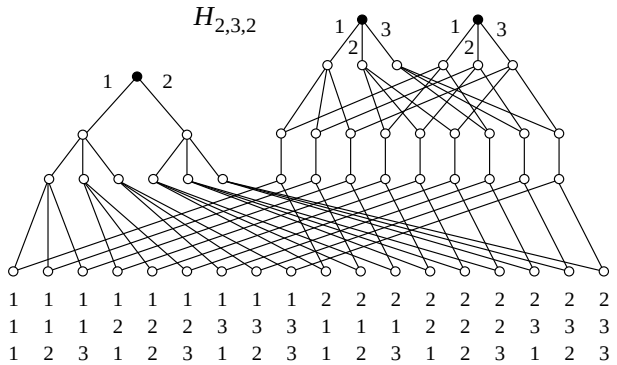

Figure 5: The $H_{p,\delta,k}$ graph, for $p=2$, $\delta=3$ and $k=2$.

Lemma 5 *For every triple of integers (p,δ,k), let $\mathcal{A}_{p,\delta,k}$ be the set of $(2p+2)\times(k\delta^p)$ integers matrices of the form $\begin{bmatrix} A_{p,\delta,k} \\ A \end{bmatrix}$ where $A \sim A_{p,\delta,k}$. Then, for every $p \geq 2$ and for every $\delta \geq k \geq 2$,*

$$\frac{(k\delta^p)!}{(k!)^2(\delta!)^{2p}(2p+2)!} \leq |\mathcal{A}_{p,\delta,k\,/\sim}| = 2^{\Omega(k\delta^p \log(k\delta^p))}$$

Proof. Similar to the proof of Lemma 3. □

Theorem 1 *Let ε, $0 < \varepsilon < 1$, be a fixed constant. For every integer d, $3 \leq d \leq \varepsilon n$, there exists a graph of order n and maximum degree d that has a memory requirement of $\Theta(n \log d)$ bits on $\Theta(\log_d n)$ vertices.*

Proof. Let $\varepsilon \in]\,0,1[$ be any fixed constant. Let n be a sufficiently large integer and let d be any integer such that $3 \leq d \leq \varepsilon n$. Similarly to the proof of Proposition 1, we build a graph of constraints of order n and maximum degree d. This graph is also denoted by $G_{n,d}$ because, for small values of d, Proposition 1 can be seen as a particular case of this theorem. Our graph construction depends on the degree d.

Small degree: $3 \leq d \leq \sqrt{n/10}$

Let $\delta = d-1$ and let p the largest integer ≥ 2 such that $\lfloor n/\delta^p \rfloor \geq 10$. Let $k = \lfloor n/\delta^p \rfloor - 8$. An integer $p \geq 2$ exists, because in particular for $p=2$, $\lfloor n/\delta^p \rfloor > n/d^2 - 1 \geq n/\sqrt{n/10}^2 - 1 = 9$, and $\lfloor n/\delta^p \rfloor > 9$ implies $\lfloor n/\delta^p \rfloor \geq 10$. In sequel, $2 \leq k \leq \delta = d-1$. The $G_{n,d}$ graph is composed of two $H_{p,\delta,k}$ graphs that we connected by a perfect matching. Similarly to the proof of Proposition 1, $G_{n,d}$ belongs to a graph of constraints of $\mathcal{A}_{p,\delta,k}$ introduced in Lemma 5. It has $2p$ constrained vertices. By definition $(k+8)\delta^p = \lfloor n/\delta^p \rfloor \delta^p = n - r$, where $r = (n \bmod \delta^p) \in \{0,\ldots,n-1\}$ because $n \geq \delta^p$. Thus, $(k+8)\delta^p \leq n < 2(k+8)\delta^p$, therefore $k\delta^p = \Theta(n)$. Hence $p = \Theta(\log_\delta(n/k)) = \Theta(\log_\delta n - \log_\delta k) = \Theta(\log_d n)$, because $2 \leq k \leq \delta$ and $\log\delta = \Theta(\log d)$. As in the proof of proposition 1, we can add a path to obtain a graph of order n exactly. Applying Lemma 1 and Lemma 5, the sum of the memory requirement on $2p$ vertices is $\Omega(k\delta^p \log(k\delta^p)) - O(n) = \Omega(n \log n)$ bits.

Large degree: $\sqrt{n/10} \leq d \leq \varepsilon n$

Let $n' = \lfloor (1-\varepsilon)n \rfloor$ and $\delta = \lfloor \sqrt{n'/10} \rfloor - 1$. The construction of $G_{n,d}$ for large d consists of a $G_{n',\delta+1}$ graph, where we add a copy of a $K_{1,d-1}$ tree connected by its root with an edge to a non constrained vertices of $G_{n',\delta+1}$. $G_{n',\delta+1}$ is of degree at most $\delta+1 \leq \sqrt{n'/10} \leq d$. Clearly the maximum degree of such a graph is d, due to the root of the $K_{1,d-1}$ tree. Shortest paths between the $2p'$ constrained vertices of $G_{n',\delta+1}$ are unchanged by the $K_{1,d-1}$ added. Finally we can add, for example to a leaf of $K_{1,d-1}$, a path of length $n - (n'+d) = n - (\lfloor (1-\varepsilon)n \rfloor + \varepsilon n) \geq 0$ to obtain a graph of exactly n vertices. From the previous case, the sum of the memory requirement on $2p' = \Theta(\log_\delta n')$ vertices is $\Omega(n' \log n')$ bits. Because $n' = \Theta(n)$ and $\log\delta = \Theta(\log n) = \Theta(\log d)$, then the sum of the memory requirement on $\Theta(\log_d n)$ vertices is $\Omega(n \log n)$ bits.

Therefore, for every degree d such that $3 \leq d \leq \varepsilon n$, it follows that the sum of the memory requirement on $\Theta(\log_d n)$ vertices of the $G_{n,d}$ graph is $\Omega(n \log n)$ bits. Let $p'' = \Theta(\log_d n)$ this number of constrained vertices. Thus the memory requirement is $\Omega(n \log n)/p'' = \Omega(n \log d)$ bits on at least one constrained vertex among p''. Since $O(n \log d)$ bits are enough to route along shortest paths then $\Theta(p'') = \Theta(\log_d n)$ routers need $\Theta(n \log d)$ bits each. □

Some authors have suggested to relax the assumption of shortest paths. The *stretch factor*, introduced in [18], denotes the maximum ratio between the length of the rout-

ing path and the distance, overall pairs source-destination. In practice we are interested by near-shortest path routing schemes using a small stretch factor, because clearly routing schemes of stretch factor of $O(n)$ can be describe in a very compact way. The $\Omega(n^2)$ bits lower bound on the global memory requirement was proved for every stretch factor < 2 in [8]. In [9] a tight lower bound on the memory requirement (locally) of $\Theta(n \log n)$ bits for every stretch factor < 2, but this large memory requirement was required by at most $O(n^{\varepsilon})$ vertices (ε constant < 1) in a worst-case graph of maximum degree $\Theta(n)$. Using Theorem 1, the $G_{n,\lfloor\sqrt{n/10}\rfloor}$ graph construction allows to easily prove a tight lower bound of $\Theta(n \log n)$ bits of memory requirement on a constant number of routers, but for every stretch factor $< 5/3$ and for a maximum degree of $O(\sqrt{n})$.

4 Global Memory Requirement

In the literature concerning the compact routing problem, lower and upper bounds have been given for the global memory requirement (*e.g.* see [4, 8, 18]). Even though the local memory requirement seems to be more important in practice for the concrete design of routing chips, the global memory requirement can give a theoretical information about routing in the network.

Definition 7 (Global Memory Requirement of a Graph) *Given a graph $G = (V, E)$, the* global memory requirement of the graph G *is the value denoted by* $\mathrm{MEM}_{global}(G)$ *:*

$$\mathrm{MEM}_{global}(G) = \min_{R} \sum_{x \in V} \mathrm{MEM}(G, R, x)$$

where R is minimized over all shortest path routing functions on G, such that vertex-labels are from $\{1, \ldots, |V|\}$ and port labels of every $x \in V$ are from $\{1, \ldots, \deg(x)\}$.

In general we are interested by the local memory requirement of a router rather than the global memory requirement of the whole graph, because a global value does not gives in general the way to distribute and balance the routing information. The following result shows that from a lower bound of the local memory requirement we can derive a lower on the global memory requirement by a graph transformation.

Lemma 6 *Let $\mathcal{M}$ be a set of $p \times q$ integer matrices. Let $\mathcal{G}$ be a set of graphs of constraints of $\mathcal{M}$ such that every graph $G \in \mathcal{G}$ is of order at most n and maximum degree $d \geq 3$, and such that every constrained vertices x of G satisfies $2 \leq \deg(x) \leq n/p$. Then for every integer $n' \geq 9n$, there exists a graph G' of order n' and of maximum degree d that has a global memory requirement of $\Omega(nm/p)$ bits, where $m = \log_2 \left(|\mathcal{M}_{/\sim}| / \binom{n'}{q}\right) - O(\log n')$.* □

This lemma has an important application. Assume that a graph of constraints G from a set of matrices $\mathcal{M}$ with n vertices and maximum degree d satisfies that its p constrained vertices have degree $\leq n/p$ and a memory requirement of at least $f(n)$ bits each. Then we can split every constrained vertices (as shown in Figure 6) to construct a new graph G' of order $9n$, of maximum degree d, and with a global memory requirement of $\Omega(nf(n))$ bits.

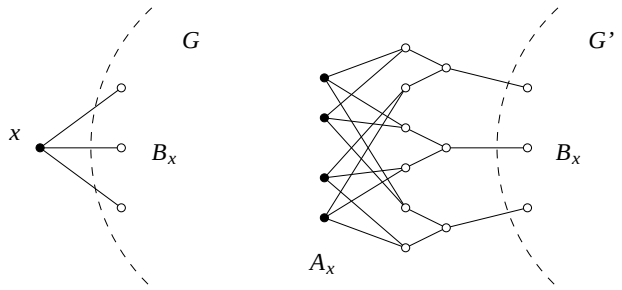

Figure 6: A constrained vertex of G split in 4 constrained vertices in G'

An application of Theorem 1 and Lemma 6 is the following corollary.

Corollary 1 *Let ε, $0 < \varepsilon < 1$, be a fixed constant. For every integer d, $3 \leq d \leq \varepsilon n$, there exists a graph of order n and maximum degree d that has a memory requirement of $\Theta(n \log d)$ bits on $\Theta(n)$ vertices.*

This result proves that the global memory requirement of a graph can be $\Omega(n^2 \log d)$ bits.

Proof. Theorem 1 shows that there exists a graph $G_{\lfloor n/9 \rfloor, d}$ of order $\lfloor n/9 \rfloor$ and maximum degree d, for every $3 \leq d \leq \varepsilon n$, ε being any fixed positive constant < 1, that has a memory requirement of $\Theta(n \log d)$ bits on $\Theta(\log_d n)$ constrained vertices. This requirement is due to a large set of $p \times q$ matrix of constraints $\mathcal{M}$ such that $\log_2 |\mathcal{M}_{/\sim}| = \Omega(n \log n)$. The maximum degree of constrained vertices is $\delta = O(\sqrt{n})$ (see the construction in the proof of Theorem 1) and moreover $n/p = \Theta(n/\log_d n) = \Omega(n/\log n)$. For a sufficient large integer n, $O(\sqrt{n}) \leq \Theta(n/\log n)$, *i.e.* $\delta \leq n/p$. Therefore we can apply Lemma 6 and build a graph, denoted Split-$G_{n,d}$, of order n and maximum degree d, that has a memory requirement of $\Omega(n \log d)$ bits on $\Theta(n)$ vertices. □

As in the end of the previous section, we can derive a lower bound of the global memory requirement for near-shortest routing schemes: $\Omega(n^2 \log n)$ bits for every stretch factor $< 7/5$ and for a worst case n-node network of maximum degree $O(\sqrt{n})$ (for more information see [13]).

5 Conclusion and Open Problems

We have built an n-node network of maximum degree d for which there is no shortest path routing scheme more compact than routing tables. Our construction is based on a random perfect matching. Moreover our network can be described with at most $O(n \log n)$ bits. In the case of large maximum degree, we prove therefore by applying Corollary 1 than the memory requirement of $\Theta(n)$ routers of our network can be $\Theta(n \log n)$ bits each. It means that there is

no efficient way to distribute (in memory space) a shortest path routing function on this network, since $\Theta(n)$ routers require to store the full description of the network.

However, our network requiring $\Theta(n^2 \log d)$ bits of routing information is based on connecting two very structured networks. It would be interesting to know if a d-regular random graph have the same property for the routing memory requirement, especially for maximum degree $o(\sqrt{n})$. More precisely: What is, on average, the memory requirement for a graph of order n and maximum degree d? It seems not easy to show that a graph of maximum degree 3 requires $\Theta(n^2)$ bits, on average, of global memory requirement, although we know that such a graph exists. Note that the average case has been studied recently in [4]. In particular, they show that for "almost all graphs" of order n $O(n)$ bits per router are necessary and sufficient for routing of shortest paths. However "almost all graphs" have a maximum degree in $\Theta(n)$ and therefore the problem stays open for degree d fixed.

The best upper bounds proposed on the compact routing problem are based on routing schemes that generate paths that are not always shortest path routing schemes, but of length close by a factor s (called the stretch factor) to the minimal possible length. Moreover these routing schemes use vertex-labels of $O(\log^2 n)$ bits size (see [1, 2, 18]). In the assumption of vertex-labels of $O(\log^2 n)$ bits size, what is the lower bound of the memory requirement?

Acknowledgments: The authors are grateful to David Peleg and Eli Upfal for helpful discussions and remarks about these results.

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

[4] 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. To appear.

[5] 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.

[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] ———, *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.

[9] P. Fraigniaud and C. Gavoille, *Universal routing schemes*. Manuscript submitted to JDC, June 1995.

[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, *Complexité mémoire du routage dans les réseaux distribués*, PhD thesis, École Normale Supérieure de Lyon, 46, allée d'Italie, Jan. 1996.

[14] 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.

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

[16] E. Kranakis and D. Krizanc, *Lower bounds for compact routing*, Tech. Rep. TR-95-18, Carleton University, July 1995. To appear in STACS '96.

[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, Utrecht, June 1993, Springer-Verlag, pp. 338–349.

[18] 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.

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

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