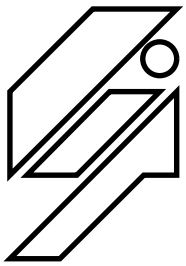


Laboratoire de l'Informatique du Parallélisme

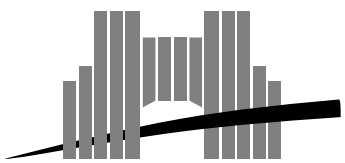
Ecole Normale Supérieure de Lyon
Unité de recherche associée au CNRS n°1398

Memory Requirement for Routing in Distributed Networks

Cyril Gavoille
Stéphane Pérennès

November 1995

Research Report N^{o} 95-??



Ecole Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : (+33) 72.72.80.00 Télécopieur : (+33) 72.72.80.80
Adresse électronique : lip@lip.ens–lyon.fr

Memory Requirement for Routing in Distributed Networks

Cyril Gavoille
Stéphane Pérennès

November 1995

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 at least $\Theta(n \log d)$ 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 cannot do better than routing tables. Moreover, we prove that, for the well-known interval routing scheme, there exists an n-node network of bounded degree d, for every $d \geq 3$, that requires $\Theta(n)$ intervals on $\Theta(n)$ links to code any shortest path routing function. This tight lower bound shows that, for networks of bounded degree, the interval routing scheme can be worst than the routing tables in term of size.

Keywords: communication on parallel and distributed networks, compact routing tables, interval routing, shortest path routing

Résumé

Dans cette étude nous nous intéressons au routage compact dans les réseaux disrtribués, c'est à dire à l'implémentation de fonctions de routage par des plus courts chemins dont le codage soit minimal sur chaque routeur. Nous construisons des graphes de taille n et de degré borné par $3 \leq d \leq \varepsilon n$ pour une quelconque constante $0 < \varepsilon < 1$, telle que la capacité mémoire nécessaire afin de router soit, sur une fraction constante des routeurs, d'au moins $\Theta(n \log d)$ bits. Ceci signifie que sur ces réseaux, quelque soit le mode de routage (par intervalles, booléen, par préfixes, ...), certains routeurs devront mémoriser quasimment la table de routage in extenso. Enfin, ces graphes sont utilisés pour montrer qu'il existe des réseaux de degré borné par 3 où le routage par intervalles utilise $\Theta(n)$ intervalles sur $\Theta(n)$ arcs, ce qui requière plus de mémoire que les tables de routage.

Mots-clés: communication sur réseaux parallèles et distribués, tables de routages compactes, routage par intervalles, routages de plus courts chemins

Memory Requirement for Routing in Distributed Networks*

Cyril Gavoille†
gavoille@lip.ens-lyon.fr
LIP, CNRS
École Normale Supérieure de Lyon
69364 Lyon cedex 07
France

Stéphane Pérennès‡
sp@essi.fr
I3S, CNRS
Université de Nice-Sophia Antipolis
06903 Sophia Antipolis
France

November 8, 1995

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 at least $\Theta(n \log d)$ 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 cannot do better than routing tables. Moreover, we prove that, for the well-known interval routing scheme, there exists an n-node network of bounded degree d, for every $d \geq 3$, that requires $\Theta(n)$ intervals on $\Theta(n)$ links to code any shortest path routing function. This tight lower bound shows that, for networks of bounded degree, the interval routing scheme can be worst than the routing tables in term of size.

1 Introduction

1.1 The Compact Routing Problem

The *general routing problem* in a network consists in 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. XY-routing [5] or e-cube routing [6] 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 [8, 10], and finding paths minimizing congestion can be done easily for a large set of networks (namely Cayley graphs) [22]. 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

*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.

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 [28, 16, 17]). Among them, *interval routing* [28, 29, 3, 11] was chosen by *Inmos* for its C104 routing chip [7]. Unfortunately, it was proved in [24, 9, 11] 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 [20] 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 [12].

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 (for instance $O(\log n)$ memory bits for the shortest path 1-interval routing scheme on bounded degree networks), but they do not apply on all networks. In the last PODC Symposium, in [13], it is proved that any universal routing scheme on n-node network, requires a total of $\Omega(n^2)$ bits of memory to store a shortest path routing function. In fact, the worst cases networks that require the total of $\Omega(n^2)$ bits of memory have a very large degree, namely $\Theta(n)$, and that is not a realistic hypothesis for real networks.

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. 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 [27, 13, 23]. A straightforward consequence of this result is that there exists an n-node network of maximum degree 3 that requires $\Omega(n/\log n)$ intervals on almost all the links to code a shortest path routing function using the interval routing scheme. This already improves by a factor $\log n$ the lower bounds established in [4, 19]. More fundamentally we give a tight lower bound of $\Theta(n)$ intervals on $\Theta(n)$ links for an n-node network of bounded degree $d \geq 3$. That shows that, in the worst case, the interval routing scheme can be worst than the routing tables, because the intervals are stored extensively by using $\Theta(n \log n)$ bits.

In the next section we describe the model of routing functions that is considered in this paper. In Section 1.3, we introduce some definitions about the *memory requirement* of a routing function. In Section 1.4, we summarize the results. In section 2, we give some technics and definitions to prove lower bounds on the memory requirement. We state our main result in Section 3. Section 4 gives a general result that allows to increase the number of nodes that requires the critical large amount of routing information. In Section 5 we consider the particular case of the interval routing scheme. Finally, we give several in the last section.

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.

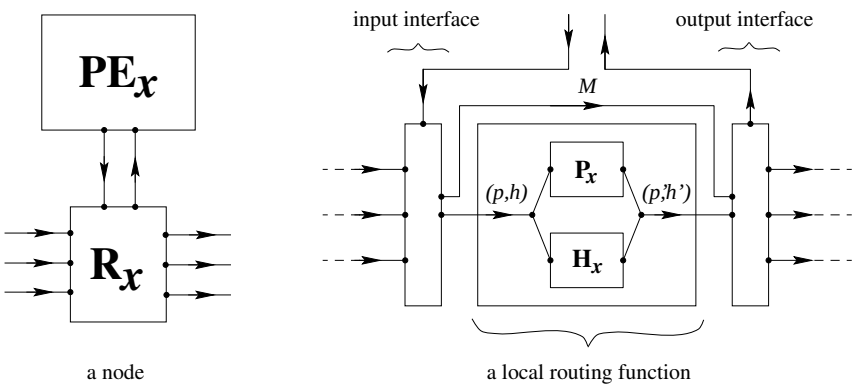


Figure 1: A model of router

In a sense, finding a routing function in a symmetric digraph $G = (V, E)$ consists to label vertices and arcs of G such that the local routing function can compute a path between any pair source-destination of vertices. 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, connected to a same end-point vertex or to a same start-point vertex, have two distinct labels. It means that we label by the same integer an output link and input link of a same edge. The collection $\mathcal{E} = \{\mathcal{E}_x \mid x \in V\}$ 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 is 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 [27] (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 $\mathcal{E} = \{\mathcal{E}_x \mid x \in V\}$ satisfying: for every couple (x, y) of vertices of G, there exist:*

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

such that:

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

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 built by R is 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 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 $\{R_x \mid x \in V\}$ such that $R_x(p, h) = R(x, p, h)$, for every $x \in V$, and every $(p, h) \in I\!N^2$. 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 $(p, h) \in I\!N^2$.

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 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 [27]. 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 finction only. Tradeoffs between memory requirement and length of the routing path have been studied in [27, 1, 2]. 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 (see [14] for more precisions about lower and upper bounds on memory requirement).

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 [26]. 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 support 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 forces the *PE*s to know what are vertex-labels available in the network, that may waste many bits in the inner *PE*'s memory. It is why we will limit the class of routing functions that we consider to those that make use of a vertex-labeling from $\{1,\ldots,n\}$ and a local arc-labeling from $\{1,\ldots,\deg(x)\}$, where $\deg(x)$ the degree of x.

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 neigbor 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 $R' = \{R'_x = (P'_x, H'_x) \,|\, x\in V\}$ 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. 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 in 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 [28], outerplanar graphs [15] and all unit circular-arc graphs [12] 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*. Some authors have considered the sum of the routing information distributed in the network, $\mathrm{MEM}_{global}(G) = \min_R \sum_x \mathrm{MEM}(G,R,x)$. Indeed routing schemes are not always evenly balanced among the vertices. For example trees and outerplanar graphs require a total of $O(n\log n)$ memory bits, since they have $O(n)$ arcs in total.

To prove lower bounds for the memory requirement, we made the model as general as possible. We summarize bellow 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 cost on the arcs.

(4) Vertex are labeled by integer in $\{1, \ldots, n\}$.

(5) For every vertex x, port of x are labeled by integer in $\{1, \ldots, \deg(x)\}$.

(6) Each vertex x knows its local routing function R_x and its local port labeling.

Point (3) means that the lower bound on directed and non uniform weighted graphs is valid since our bound is proved on a smallest subset of graphs. A contrary to the model of Peleg and Upfal in [27], the label of the vertex x is not necessary known by the router. Of course it is reasonable to assume that the processor PE_x has enough large memory to store its own label (to avoid to send messages to itself for instance). It is possible that some function R_x does not use the label of x as the following example suggests. Consider the complete bipartite graph $K_{m,n-m}$ of n vertices where m is a constant. We assume that headers are the labels of the destination.Thus, R_x can be defined by its port function P_x. Vertices of the first partition are labeled from $\{1, \ldots, m\}$, and vertices of the second partition are labeled from $\{m+1, \ldots, n\}$. The port between x and y is labeled $y - m$ if $x < y$ and y otherwise, that implies for every x port labels are from $\{1, \ldots, \deg(x)\}$. The routers runs the following algorithm: for every port p, for every destination y and for every router x of the first partition,

$$P_x(p, y) = \text{“}\textit{if } y = x \textit{ then } P_x(p, y) \leftarrow 0 \textit{ else } P_x(p, y) \leftarrow \max\{y - m, 1\}\text{”}$$

For every router x of the second partition,

$$P_x(p, y) = \text{“}\textit{if } y \leq m \textit{ then } P_x(p, y) \leftarrow y \textit{ else if } p = 0 \textit{ then } P_x(p, y) \leftarrow 1 \textit{ else } P_x(p, y) \leftarrow 0\text{”}$$

The memory requirement of $K_{m,n-m}$ thus is $O(1)$ bits since in the above algorithm description 0, m and every $x \in \{1, \ldots, m\}$ are constants.

Assumption (6) means that a router x can send an arbitrary message M on the physical link labeled p without memory overhead: router x knows where is the port p. This step is performed by the interfaces of x. The memory requirement does not include the memory of the interfaces. However it is well known that using the electronic technology the interfaces of a router x needs $O(\log \deg(x))$ bits (send M on $\deg(x)$ possible wires is equivalent to store in a register the address of the port). But this memory bits are dynamic memory (RAM) that is different kind of memory bits defined in the memory requirement which can be seen as ROM bits. Moreover technology of interfaces may change.

We have seen in the first example that assumption (4) on the size of the vertex-labels is important. We will see in Section 3 that if the n vertex-labels would have been taken in the set $\{1, \ldots, n^2\}$, we could not have apply our arguments. However assumption (4) is implicitly used in the literature (*e.g.* see [27] and [13]), even if some upper bounds use longer labels (see [27, 16, 17, 1, 2]).

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 $\Omega(n \log n)$ bits thus $\Omega(n \log d)$ each, 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 will prove a general result that allows to increase the number of routers having to know $\Theta(n \log d)$ 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 bellow a table that summarizes bounds on the memory requirement, and the number of routers using such a requirement as function of the maximum degree d and the order of the graph. The last column of the table is the sum of the memory requirement on all the routers of 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 [29]). In this table, ε is an arbitrary positive constant < 1, and m an integer constant.

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

$^{(*)}$This example is under the assumption that each PE does not send message to itself.

2 Matrices and Graphs of Constraints

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 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.

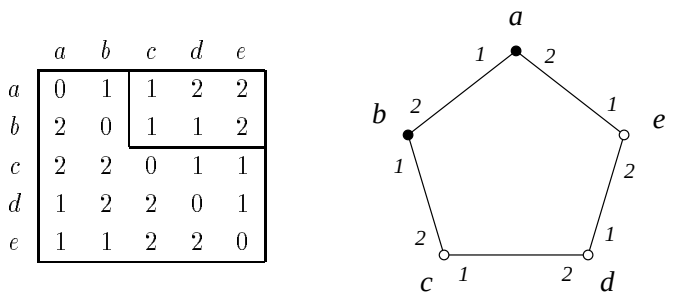
Recall the notion of matrix and graph of constraints introduced in [14]. The basic idea is to apply a counting argument to a set of graphs that computes a finite set of integer matrices.

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. Let us consider the ring C_5 of vertices a, b, c, d, e, where ports are labeled from $\{1, 2\}$. Figure 2 shows a routing table that codes a shortest path routing function on C_5. The framed 2×3 sub-matrix is a matrix of constraints from the constrained vertices $\{a, b\}$ to the target vertices $\{c, d, e\}$. Indeed, the ports of constrained vertices $\{a, b\}$ can be labeled by integers 1 or 2 according the matrix of constraints such that every shortest path reaching a target vertex starts in using this port. Note that entries of a matrix of constraints can not contain 0, since entries of the matrix are arc-labels that, by definition, are from $\{1, \ldots, \deg(x)\}$, for a vertex x.



	a	b	c	d	e
a	0	1	1	2	2
b	2	0	1	1	2
c	2	2	0	1	1
d	1	2	2	0	1
e	1	1	2	2	0

Figure 2: An example of matrix of constraints for a ring

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.

From the above definitions, M and M' are two matrices of constraints of G for same set of constrained vertices and the same set of target vertices, if and only if $M' \sim M$. 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.*

Note that $|\mathcal{M}_{/\sim}|$ is the number of nonequivalent matrices of constraints for a set of graphs of constraints of $\mathcal{M}$.

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.

Proof. Let (p, q) be any pair of integers, and let $\mathcal{M}$ be a set of $p \times q$ integer matrices. Let $\mathcal{G}$ be a set of graphs of constraints of $\mathcal{M}$. 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 R 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 $f_B : \{1, \ldots, n\} \to \{0, 1\}$, such that $f_B(x) = 1$ if and only if there exists a vertex $b \in B$ that is labeled x. Let us show that the knowledge of R_A, f_B and $O(\log n)$ bits is sufficient to compute a matrix $M' \sim M$, that is to determine to which set of $\mathcal{M}_{/\sim}$ M belongs. Assume that an observer knows only R_A, f_B and the following procedure:

"Arrange the set of the port functions of R_A in an arbitrary order $\{P_1, \ldots, P_p\}$.
Set $j \leftarrow 0$.
For i from 1 to p do
  for b from 1 to n do
    if $f_B(b) = 1$ then set $j \leftarrow j + 1$
      set $M'_{i,j} \leftarrow P_i(0, b)$"

Clearly the observer has enough information to compute a matrix M', which by definition of the graph of constraints is $\sim M$. Let m the minimum number of bits to necessary store R_A. The procedure itself, excluding f_B and the P_i's functions, can be stored with at most $O(\log n)$ bits (we need to know p, q and n which represent at most $O(\log n)$ bits). Moreover every characteristic function f_Y of a subset $Y \subset X$ — $f_Y(x) = 1 \Leftrightarrow x \in Y$ — can be stored with at most $\log_2 \binom{|X|}{|Y|} + O(\log |X|)$ bits, for a finite set X. Since this above computation of M' can be done for any matrix $M \in \mathcal{M}$, we conclude, by a counting argument, that there exists some matrix $M_0 \in \mathcal{M}$ such that every matrix $\sim M_0$ requires at least $\log_2 |\mathcal{M}_{/\sim}|$ bits to be stored, and thus that $m + \log_2 \binom{n}{q} + O(\log n) \geq \log_2 |\mathcal{M}_{/\sim}|$. Note that the test $M \sim M_0$, that is equivalent to compute a canonical representative of the class of M, can be done by an algorithm depending of p and q only, *i.e.* of size at most $O(\log n)$ bits. Therefore the sum of the memory requirement on p routers is at least m bits where $m \geq \log_2(|\mathcal{M}_{/\sim}| / \binom{n}{q}) - O(\log n)$. □

3 Graph Constructions

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 arbitrary degree $3 \leq d \leq \varepsilon n$, where ε is a fixed positive constant < 1.

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 $a_{i,j} = (\lfloor (j-1)/\delta^{i-1} \rfloor \bmod \delta) + 1$, 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*

Proof. Let (p,δ) a pair of integers such that $p \geq 1$ and $\delta \geq 2$. The graph $H_{p,\delta}$ is defined by induction on p. $H_{1,\delta}$ is isomorph to $K_{1,\delta}$. Clearly (1), (2) and (3) are satisfied since the matrix $A_{1,\delta} = [1\,2\,\ldots\,\delta]$ is a matrix of constraints for $H_{1,\delta}$. Precisely $A_{1,\delta}$ is the matrix of constraints with a constrained vertex, labeled 1, forming a singleton $A_1 = \{1\}$ that is the root of $K_{1,\delta}$, and with the target vertices forming a set $B_1 = \{1,\ldots,\delta\}$ that are the leaves of $K_{1,\delta}$.

Let us now assume that $H_{p,\delta}$ possesses a set of constrained vertices A_p labeled $\{1,\ldots,p\}$, and a set of target vertices B_p labeled by all the words of length p from the alphabet $\{1,\ldots,\delta\}$. Moreover, assume that properties (1), (2) and (3) are satisfied. In the following we denote by $T_{p,\delta}$, the complete δ-ary tree of height p and of δ^p leaves. The order of $T_{p,\delta}$, denoted by $|T_{p,\delta}|$, is $(\delta^{p+1}-1)/(\delta-1)$. $H_{p+1,\delta}$ 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, as shown on the left side of Figure 3. For the particular case $p+1=2$, we do not add the first $\delta^p = \delta$ intermediate vertices. The root of $T_{p,\delta}$ is labeled $p+1$. It forms with the set A_p, the new set of constrained vertices $A_{p+1} = A_p \cup \{p+1\}$. Each 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. 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 u 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}. Therefore every pair (u,x), where $u \in B'_p$ and x is a leaf of $T_{p,\delta}$, is connected by a unique vertex $w = ui \in B_{p+1,\delta}$, if x is labeled by i (see the right side of Figure 3).

Note that, on Figure 3, and according the labeling that we used, the shortest paths leading to the target vertex $xyz \in B_3$ have to traverse 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.

The degree of $H_{p,\delta}$ is $\delta+1$, due to the leaves of $T_{p,\delta}$. This proves (1). The order of $H_{p+1,\delta}$, is $|H_{p+1,\delta}| =^1 |H_{p,\delta}| + |T_{p,\delta}| + \delta^p + \delta^{p+1} = |H_{1,\delta}| + \sum_{i=1}^{p}(|T_{i,\delta}| + \delta^i + \delta^{i+1}) = 1 + \delta + \sum_{i=1}^{p}((\delta^{i+1}-1)/(\delta-1) + \delta^i + \delta^{i+1}) = 1 + \delta + \delta(\delta^p-1)(\delta^2+\delta-1)/(\delta-1)^2 - p/(\delta-1) \leq 5\delta^{p+1}$, for every $p \geq 1$ and $\delta \geq 2$, proving (2). Considering the constrained vertices A_{p+1} and the target vertices B_{p+1}, the matrix $A_{p+1,\delta}$ is a matrix of constraints of $H_{p+1,\delta}$ because every vertices $w = ui \in B_{p+1}$ is connected by an unique shortest path to B'_p. Thus, by hypothesis, every vertex $w \in B_{p+1}$ is connected to any constrained vertices of A_p. Moreover, w is also connected by an unique shortest path to the i^{th} child of $T_{p,\delta}$. This proves yields (3) and completes the proof. □

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 $A_{p,\delta}$,

[1] According to Figure 3, for $p \geq 2$ we can remove δ intermediate vertices, that does not disturb the final result in magnitude order.

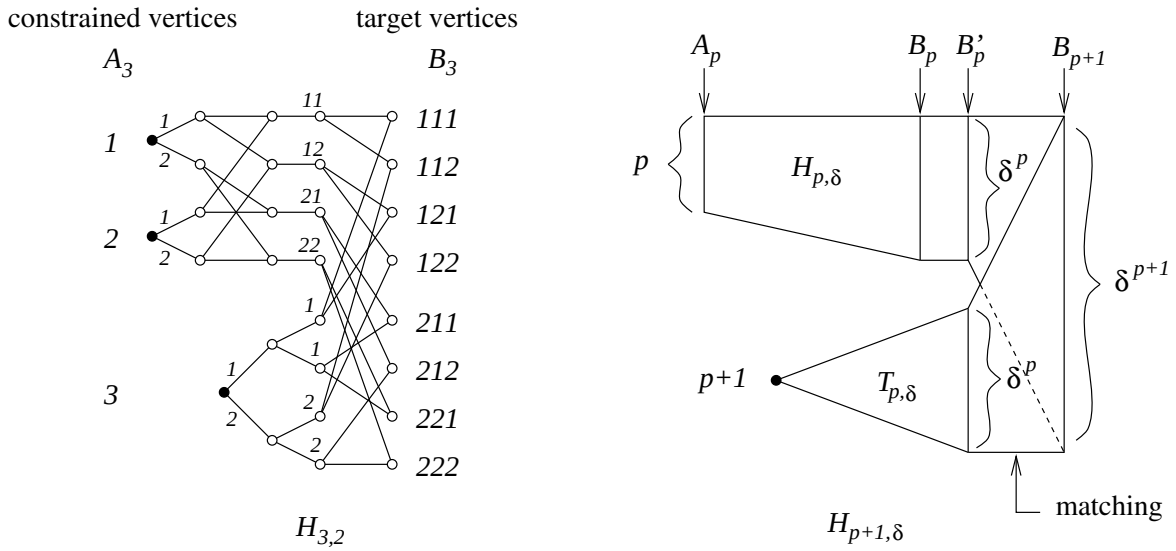


Figure 3: The $H_{p,\delta}$ graph. The shortest paths leading to the target vertex $xyz \in B_3$ have to use the port x from the constrained vertex $1 \in A_3$, the port y from the vertex 2 and the port z from the vertex 3.

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 $|\mathcal{A}_{/\sim}| \geq \delta^p!^2/((\delta!)^{2p}(2p)!)$, where $(\delta!)^{2p}(2p)!$ is the maximum size of a set of $2p \times \delta^p$ integer matrices from $\{1,\ldots,\delta\}$ 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 $p \geq 2$, $L = \Omega(p\delta^p \log\delta) = \Omega(\delta^p \log\delta^p)$. □

Proposition 1 *Let α be 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.*

For maximum degree $\Theta(n^{1/3})$ this proposition already proves that there exists a vertex that has a memory requirement of $\Theta(n \log n)$ bits, though for particular values of n.

Proof. Let n be a sufficiently large integer of the form $n = \alpha(d-1)^p$, where α is a constant larger than 10 and p is an integer larger than 2. Let d be any integer satisfying $3 \leq d \leq n^{1/3}$. 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. Roughly speaking, applying Lemma 1 and Lemma 3, the sum of the memory requirement on these $2p$ vertices is $\Omega((d-1)^p \log(d-1)^p) - O(n)$, 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 $d \geq 3$.

More precisely let $\delta = d-1$ and let $p = \log_\delta(n/\alpha)$. Also, let M be a matrix of $\mathcal{A}_{p,\delta}$. Let H_1 and H_2 be two copies of the $H_{p,\delta}$ graph (defined in Lemma 2) of constrained vertices A_1, A_2 and of target vertices B_1, B_2 respectively. Such graphs exist because $\delta \geq 2$ and $\delta^2 \leq n^{2/3} \leq n/\alpha$ imply $2 \leq \log_\delta(n/\alpha) = p$. The graph is composed of H_1 and H_2 connected by a perfect matching between B_1 and B_2 (see Figure 4), plus few additional vertices eventually. More precisely $w_1 \in B_1$ is connected to $w_2 \in B_2$ if and only if the vector $w_1 w_2$ formed by the concatenation of the words w_1 and w_2 is a column of M (recall that vertices of B_1 or B_2 are labeled by words of length p on the alphabet $\{1,\ldots,\delta\}$). By construction of $H_{p,\delta}$, vertices A_1 are constrained for vertices $B_1 \cup B_2$, because B_1 and B_2 are connected by a unique edge of the perfect matching.

By symmetry, A_2 is also constrained vertices for $B_1 \cup B_2$. Finally $A_1 \cup A_2$ are constrained vertices for the set B_1. One can check that M is a matrix of constraints of constrained vertices $A_1 \cup A_2$ on the target vertices B_1. From Lemma 2, the order of such construction is m where $m \leq 10\delta^p \leq \alpha\delta^p = n$. We add a path of length $n - m$ to any vertex, but a constrained one, in order to obtain a graph of order exactly n. By hypothesis, $\delta^p = n/\alpha = \Theta(n)$ and $p = \Theta(\log_\delta n) = \Theta(\log_d n)$, for $d \geq 3$. Since $p \geq 2$ and $\delta \geq 2$, we can apply Lemma 3, and by application of Lemma 1, there exists a graph, denoted by $G_{n,d}$, such that the sum of the memory requirement on $2p$ constrained vertices is at least $\Omega(\delta^p \log \delta^p) - O(n) = \Omega(n \log n)$ bits. Thus, the memory requirement of $G_{n,d}$ is $\Omega(n \log n)/(2p) = \Omega(n \log d)$ bits on at least one vertex. Since this memory requirement is at most $O(n \log d)$, performed by the routing tables, it follows that, in fact, $\Theta(p)$ vertices, *i.e.* $\Theta(\log_d n)$ vertices, have a memory requirement of $\Theta(n \log d)$ bits each. □

Note that the result also is true for a maximum degree $3 \leq d \leq n^{1/2-\varepsilon}$ for every constant $0 < \varepsilon < 1/2$. Whatever Theorem 1 will prove a stronger result in the next paragraph.

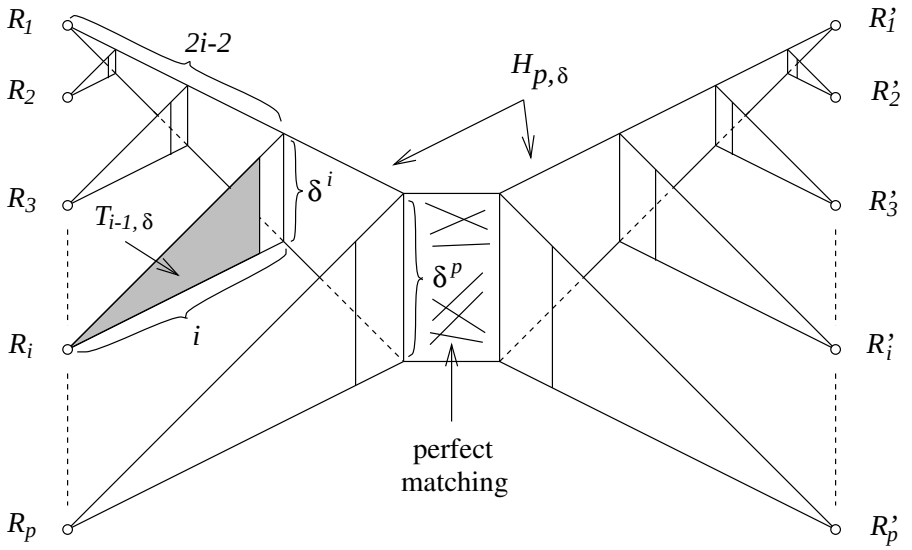


Figure 4: The $G_{n,d}$ graph. A graph of constraints of order n and of maximum degree $d = \delta + 1$. Routers R_1,...,R_p and R'_1,...,R'_p "compute" the perfect matching between $2\delta^p = \Theta(n)$ vertices.

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 $A_{p,\delta,\delta} = A_{p+1,\delta}$. For example

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

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*

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 $Q_{p,\delta,k}$ where the 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 $k\delta^p$ vertices (that are the constrained 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 $Q_{p,\delta,k}$. The order of $Q_{p,\delta,k}$ is $k(\delta^p-1)/(\delta-1)+1$.

Connections between $H_{p,\delta}$ and $Q_{p,\delta,k}$ are as follows: among the $k\delta^p$ vertices of $H_{p,\delta,k}$, every target vertex is connected exactly to one constrained vertex of $H_{p,\delta}$ and to one leave of $Q_{p,\delta,k}$. Every target vertex of $H_{p,\delta}$ is connected to k different target vertices of $H_{p,\delta,k}$. Every leaf of $Q_{p,\delta,k}$ is connected with δ target vertices of $H_{p,\delta,k}$ (see Figure 5). The order of $H_{p,\delta,k}$ is $|H_{p,\delta}|+|Q_{p,\delta,k}|+\delta^p+k\delta^p \leq 5\delta^p+k(\delta^p-1)/(\delta-1)+1+(k+1)\delta^p \leq \delta^p(k+6+k/(\delta-1))+1-k/(\delta-1) \leq (k+8)\delta^p$, for every $\delta \geq k \geq 2$ and $p \geq 2$. $A_{p,\delta,k}$ is also a matrix of constraints of $H_{p,\delta,k}$ with a similar analyze than in the proof of Lemma 2, because the target vertices of $H_{p,\delta,k}$ can be labeled by words of length $p+1$ where the first symbol is from the alphabet $\{1,\ldots,k\}$ and the p last from $\{1,\ldots,\delta\}$. Figure 5 shows a $H_{2,3,2}$ graph that possesses the matrix $A_{2,3,2}$ (put on page 12) as matrix of constraints. □

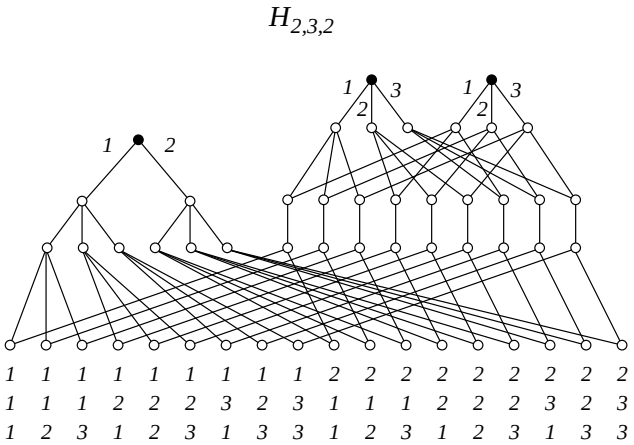


Figure 5: The $H_{p,\delta,k}$ graph, for $p=2$, $\delta=3$ and $k=2$

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. □

Remark. $|\mathcal{A}_{p,\delta,k/\sim}| = 2^{\Omega(k\delta^p \log(k\delta^p))}$ also for $p=1$ and for every $\delta \geq k \geq 3$.

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. We also denote it 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 $\log\delta = \Theta(\log d)$. 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 in 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. □

Remarks. The diameter of $G_{n,d}$ is $4p - 3 = \Theta(\log_d n)$. It is easy to verify by induction that the number of edges is less than $3n$ and its *thickness* is exactly 2 (clearly $G_{n,d}$ can not be planar because the perfect matching between the two $H_{p,\delta}$ copies). Recall that the thickness of a graph G is the minimum number of planar graphs whose the union is G (see Harary in [21, pages 120-121]). It concerns a measure of "nearness" to planarity of G that may be important for *VLSI* layout (see also [18]). Moreover the $G_{n,d}$ graph is a *quasi* d-regular graph for every $3 \leq d \leq \sqrt{n/10}$, *i.e.* it can be transformed in a d-regular graph of order $\Theta(n)$ vertices, adding extra edges and extra vertices. This implies that we can connect together all these vertices (adding some extra vertices eventually) without changing the shortest paths between constrained vertices and target vertices.

Some authors have suggested to relax the assumption of shortest paths. The *stretch factor*, introduced in [27], denotes the maximum ratio between the length of the routing 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 [13]. In [14] 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 [27, 13]). 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 is 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 $m = \log_2 (|\mathcal{M}_{/\sim}|/\binom{n'}{q}) - O(\log n')$.*

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 construct a new graph G' of order $9n$, of maximum degree d, and with a global memory requirement of $\Omega(nf(n))$ bits.

Proof. Let $G \in \mathcal{G}$ be a graph of constraints of a $p \times q$ matrix $M \in \mathcal{M}$ such that G is of order n, of maximum degree $d \geq 3$ and such that the sum of the memory requirement on its p constrained vertices is at least $\log_2 (|\mathcal{M}_{/\sim}|/\binom{n}{q}) - O(\log n)$ bits. It exists from Lemma 1. Let $2 \leq \delta \leq n/p$ be the maximum degree of the p constrained vertices of G, and let x be a constrained vertex. Let us show that, for every integer $k \geq 1$, we can *split* x in k vertices that have all the same constraints than x, with at most $6k + 3\delta/2 - 1$ intermediate vertices added and keeping the maximum degree d of G.

Let $\alpha = \lceil k/(\delta - 1) \rceil \geq 1$ and let B_x the set of the neighbors of x. We modify the graph G as follows (see the example of Figure 6): remove x (including its adjacent edges) and, to each vertices of B_x, connect an edge to the root of a tree such that, the tree is a $(\delta-1)$-ary tree (or binary tree if $\delta = 2$) of α leaves which are all at the same distance from the root. Lastly connect a set A_x of k vertices more to the leaves of such trees such that each vertex of A_x is connected to exactly $\deg(x)$ different leaves (one for each trees) and such that each leaf is connected at most to $\delta - 1$ vertices. It is always possible because the k vertices of A_x are all connected to exactly one leaf among α leaves of a tree. Therefore the maximum number of edges connecting a leaf to some vertices of A_x is $\lceil k/\alpha \rceil = \lceil k/\lceil k/(\delta-1) \rceil \rceil \leq \lceil k/(k/(\delta-1)) \rceil$, that is less than $\delta - 1$. Hence leaves are of degree at most $\delta \leq d$ (if $\delta = 2$, degree of such leaves can be $3 \leq d$ by assumption). The degree of the other vertices is clearly $\leq d$. Also there exists a unique shortest path between the k vertices of

A_x to every vertex of B_x. The number of vertices of each tree is at most $5\alpha/2-1$ ($\alpha \geq 1$ and this number is less than $2 \cdot 2^{\lfloor \log_2 \alpha \rfloor} - 1 + \lfloor \log_2 \alpha \rfloor + 1 \leq 5\alpha/2 - 1$). Therefore number of added vertices for x is at most $-1 + \delta(5\alpha/2-1) + k \leq 5\delta(k/(\delta-1)+1)/2 - \delta + k - 1 \leq 6k + 3\delta/2 - 1$.

Now we apply this splitting transformation to the p constrained vertices of G with $k = \lceil n/(6p) \rceil \geq 1$. If the graph obtained does not contain vertices of degree two, we choose a block of split vertices from a vertex x and we remove the edges connecting B_x to the root of trees, replacing these edges by a path of length two, creating therefore $\deg(x) \leq \delta$ vertices of degree two. At this step the order of the construction is $m \leq n + p(6k+3\delta/2-1) + \delta \leq n + p(6(n/(6p)+1)+3\delta/2-1) + \delta \leq 2n + 3n/2 + 5p + n/p \leq 7n/2 + 5n \leq 9n \leq n'$ ($5p + n/p$ is maximum for $p = n$). Since there exists a vertices of degree two we can add a path of length $n' - m \geq 0$ in order to obtain a graph G' of order n' and of maximum degree d.

By construction there is a unique shortest path between vertices of A_x and vertices of B_x. Therefore every vertex of A_x has the same constraints than the original constrained vertex x in G. We can choose $k = |A_x| = \Theta(n/p)$ disjoint sets of vertices $C_1, \ldots, C_k$ of p vertices each one, such that every vertex of every set A_x belongs to a different set C_i, for every constrained vertices x of G and every $i \in \{1, \ldots, k\}$. Every set C_i, $i \in \{1, \ldots, k\}$, is a set of p constrained vertices of the matrix M, thus by application of Lemma 1 the sum of memory requirement on these p vertices of C_i is at least $m = \log_2 (|\mathcal{M}_{/\sim}|/\binom{n'}{q}) - O(\log n')$ bits. Since C_i's are independent, then k blocks compute independently at least m bits, *i.e.* a total of at least $km = \Omega(nm/p)$ bits. □

Remark. Splitting the constrained vertices of a graph G of order n in k vertices provides a new graph Split-G of order $\Theta(n)$ but with $\Theta(kn)$ added edges, that may increase the thickness of the graph construction. Moreover the distance between the split constrained vertices and its initial neighbors in G is raised of $\Theta(\log_\delta k)$, where δ is the maximum degree of the constrained vertices.

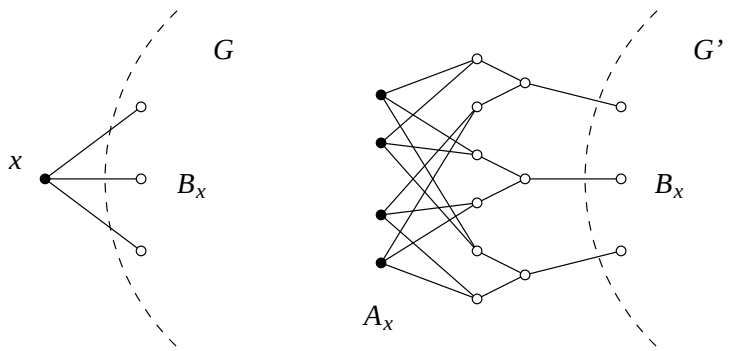


Figure 6: A constrained vertex of G split in 4 constrained vertices in G'

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 $\Theta(n^2 \log d)$ 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. □

Remark. In the Split-$G_{n,d}$ construction we use a splitting of constrained vertices of the $G_{\lfloor n/9 \rfloor,d}$ graph in $k = \Theta(n/p) = \Theta((n/\log n)\log d)$ new constrained vertices. The remarks of Lemma 6 and Theorem 1 imply that the diameter of Split-$G_{n,d}$ is $\Theta(\log_d n)$, just as the diameter of $G_{n,d}$. However its thickness[2] becomes $\Omega(k)$, and thus can be in order of n for some d.

5 Tight Lower Bound for the Interval Routing Scheme

A well-known compact routing scheme used to implement routing functions is the *interval routing scheme* introduced by Santoro and Khatib [28]. Given a routing function R on a graph G, R is coded by the interval routing scheme if vertices are labeled from $\{1, \ldots, n\}$ and, for every arc e, the set of the destination's labels traversing e is grouped in intervals. Interval means set of consecutive integers modulo n. The measure of the efficiency of this scheme for given a graph G is called the *compactness* of G and is usually defined as the maximum number of intervals that one can find on each arc, minimized over all shortest routing functions on G. Finding the compactness of G consists in finding the appropriate vertex-labeling on G. The problem of finding the worst case of compactness for graphs of order n has been addressed in [24, 9, 20] and in [4, 19, 23] for graphs of bounded degree. Here we give a general lower bound for every degree d, and we give a tight lower bound for the compactness of graphs of maximum degree bounded by a constant.

Theorem 1 showed that the best shortest path routing scheme on graphs of order n and maximum degree bounded by a constant is the routing table that uses $\Theta(n)$ bits per router. Therefore a crude lower bound on the compactness is to remark: since one interval of destination labels can code at most $O(\log n)$ bits of information, at least one arc of the graph needs of $\Omega(n/\log n)$ intervals. The following theorem proves that this lower bound is rough, albeit $\Omega(n/\log n)$ is already the recent lower bound given in [23].

Theorem 2 *Let G be a graph of order n such that p vertices of degree at most d have a memory requirement at least of m bits each. The compactness of G is at least k intervals per arc on p arcs such that*

$$\frac{m}{d} < 2k \log_2 \left(\frac{n}{2k} + 1 \right) + 2k$$

Proof. Assume that the memory requirement of x, a vertex of degree at most d in a graph G of order n, is at least m bits. Let us suppose that x uses the interval routing scheme. Let $I = [a_1, b_1] \cup \ldots \cup [a_k, b_k]$ be the set of vertex-labels from $\{1, \ldots, n\}$ associated to an arc of x, coded by at least k intervals. We can imagine that x codes the routing information with intervals, and uses all the information contained in I in a non usual way. Let us denote by $f(n,k)$ the minimum number of bits to encode the set I. We can represent the set I by a list $(d_1, \ldots, d_{2k})$ of $2k$ integers such that $d_1 = a_1$, $d_{2i} = b_{2i} - a_{2i}$, for $i \in \{1, \ldots, k\}$, and $d_{2i+1} = a_{2i+1} - b_{2i}$, for every $i \in \{1, \ldots, k-1\}$. In other words I can be coded by an offset $d_1 = a_1$ and a list of $2k-1$ lengths. Thus the list $(d_1, \ldots, d_{2k})$ may be encoded by at most $\sum_{i=1}^{2k} \lceil \log_2 d_i \rceil$ bits. Thus $f(n,k) < \log_2(\Pi_{i=1}^{2k} d_i) + 2k$. Since $\sum_{i=1}^{2k} d_i = n$, then $\Pi_{i=1}^{2k} d_i < \lceil n/(2k) \rceil^{2k}$. Thus $f(n,k) < 2k \log_2(n/(2k)+1) + 2k$. By definition of the memory requirement, $m/d \leq f(n,k)$, because at least one arc of x has a memory requirement $\geq m/d$ bits and $\leq f(n,k)$ bits. Therefore p vertices of G have at most k intervals per arc on at least one arc such that $m/d \leq f(n,k)$, that yields the result. □

Remark. In the above proof, we use mainly the fact that I can be coded by roughly $\log_2 \binom{n}{2k}$ bits, that is significantly less than $k \log n$ when k is large, using the inequality $\binom{x}{y} < (xe/y)^y$ (see in [25, page 165]).

Corollary 2 *For every constant integer $d \geq 3$, there exists a graph of order n and maximum degree d with a compactness of $\Theta(n)$ intervals per arc on $\Theta(n)$ arcs.*

[2] The thickness of every $G = (V, E)$ is at least $\lceil |E|/(3|V| - 6) \rceil$, since an n-vertex planar graph has at most $3n - 6$ edges.

We would derive a more general corollary than the corollary 2, that is, for any fixed constant ε, $0 < \varepsilon < 1$, and for every $3 \leq d \leq \varepsilon n$, there exists a graph of order n and maximum degree d (the Split-$G_{n,d}$ graph) that has a compactness of k on $\Theta(n)$ arcs, such that $k \log(n/k) = \Omega(n(\log d)/d)$.

Proof. Apply Theorem 2 for d constant and for the Split-$G_{n,d}$ graph that has a memory requirement of $\Theta(n)$ bits on $\Theta(n)$ vertices. The inequality of Theorem 2 states that $k \log(n/k) = \Omega(n)$, that implies $k = \Omega(n)$ intervals per arc on $\Omega(n)$ arcs (to be convince take $k = n/f(n)$ with $f(n)$ a function as n and show that $f(n)$ necessary is bounded by a constant). Recall that for every graph the compactness is at most $O(n)$ intervals on all its arcs and that for any bounded degree graph the number of its arcs is $O(n)$. □

6 Conclusion and Open Problems

We have built a 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 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.

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, 27]). 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] V. Braume, *Theoretische und experimentelle analyse von intervall-routing algorithmen*, Master's thesis, Department of Mathematics and Computer Science, University of Paderborn, 1993.

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

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

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

[8] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numerische Mathematik, 1 (1959), pp. 269–271.

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

[10] R. W. Floyd, *Shortest path*, Comm. ACM, 5 (1962), p. 345.

[11] P. Fraigniaud and C. Gavoille, *Interval routing schemes*, Research Report 94-04, Laboratoire de l'Informatique du Parallélisme, LIP, École Normale Supérieure de Lyon, 69364 Lyon Cedex 07, France, Jan. 1994.

[12] ——, *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.

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

[14] ——, *Universal routing schemes*, Research Report, Laboratoire de l'Informatique du Parallélisme, LIP, École Normale Supérieure de Lyon, 69364 Lyon Cedex 07, France, 1995. In preparation.

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

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

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

[18] M. R. Garey, D. S. Johnson, and H. C. So, *An application of graph coloring to printed circuit testing*, IEEE Trans. Circuits and Systems, CAS-23 (1976), pp. 591–599.

[19] C. Gavoille and E. Guévremont, *On the compactness of bounded degree graphs for shortest path interval routing*, in 2^{nd} Colloquium on Structural Information & Communication Complexity (SIROCCO), Carleton University Press, June 1995.

[20] ——, *Worst case bounds for shortest path interval routing*, Research Report 95-02, Laboratoire de l'Informatique du Parallélisme, LIP, École Normale Supérieure de Lyon, 69364 Lyon Cedex 07, France, Jan. 1995.

[21] F. Harary, *Graph Theory*, Addison-Wesley, 1969.

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

[23] E. Kranakis and D. Krizanc, *Lower bounds for compact routing*, Tech. Rep. TR-95-18, Carleton University, July 1995.

[24] 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.

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

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

[27] 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.

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

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