Laboratoire de l'Informatique du Parallélisme

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

Worst Case Bounds for Shortest Path Interval Routing (revised version)

Cyril Gavoille
Eric Guévremont

April 5, 1995

Research Report N^{o} 95-02

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

Worst Case Bounds for Shortest Path Interval Routing
(revised version)

Cyril Gavoille
Eric Guévremont

April 5, 1995

Abstract

Consider *shortest path interval routing*, a popular memory-balanced method for solving the routing problem on arbitrary networks. Given a network G, let $\text{IRS}(G)$ denote the maximum number of intervals necessary to encode groups of destinations on an edge, minimized over all shortest path interval routing schemes on G. In this paper, we establish tight worst case bounds on $\text{IRS}(G)$. More precisely for any n, we construct a network G of n nodes with $\text{IRS}(G) \in \Theta(n)$, thereby improving on the best known lower bound of $\Omega(n/\log n)$. We also establish a worst case bound on bounded degree networks: for any $\Delta \geq 3$ and any n, we construct a network G_Δ of n nodes and maximum degree Δ with $\text{IRS}(G_\Delta) \in \Omega(n/(\log n)^2)$.

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

Résumé

Nous considérons le problème du *routage par intervalles de plus courts chemins*, une méthode populaire et distribuée résolvant le problème du routage sur les réseaux arbitraires. Etant donné un réseau G, nous posons $\text{IRS}(G)$ le nombre maximum d'intervalles nécessaires pour coder les groupes de destinations sur une arête, minimisé sur tous les routages par intervalles de plus courts chemins sur G. Dans cet article, nous établissons des bornes très étroite sur le pire cas pour $\text{IRS}(G)$. Plus précisément, pour tout n, nous construisons un réseau G de n nœuds avec $\text{IRS}(n) \in \Theta(n)$, améliorant par conséquent la meilleur borne inférieure connue $\Omega(n/\log n)$. Nous établissons également un pire cas pour les réseaux de degré borné: pour tout $\Delta \geq 3$ et tout n, nous construisons un réseau G_Δ de n nœuds et de degré maximum Δ avec $\text{IRS}(G_\Delta) \in \Omega(n/(\log n)^2)$.

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

Worst Case Bounds for Shortest Path Interval Routing *

Cyril Gavoille
LIP-CNRS
École Normale Supérieure de Lyon
69364 Lyon cedex 07
France

Eric Guévremont
School of Computing Science
Simon Fraser University
Burnaby, B.C., V5A 1S6
Canada

April 5, 1995

Abstract

Consider *shortest path interval routing*, a popular memory-balanced method for solving the routing problem on arbitrary networks. Given a network G, let $\text{IRS}(G)$ denote the maximum number of intervals necessary to encode groups of destinations on an edge, minimized over all shortest path interval routing schemes on G. In this paper, we establish tight worst case bounds on $\text{IRS}(G)$. More precisely for any n, we construct a network G of n nodes with $\text{IRS}(G) \in \Theta(n)$, thereby improving on the best known lower bound of $\Omega(n/\log n)$. We also establish a worst case bound on bounded degree networks: for any $\Delta \geq 3$ and any n, we construct a network G_Δ of n nodes and maximum degree Δ with $\text{IRS}(G_\Delta) \in \Omega(n/(\log n)^2)$.

1 Introduction

The *shortest path routing problem* for an arbitrary network of processors is to design a uniform strategy that the router of each processor will follow upon reception of a message to decide to which of its neighboring nodes the message should be sent next such that the message arrives at its destination after passing through as few nodes as possible. The routing strategy should be simple and distributed so as to limit the costs of routing (space, time and complexity) and uniform to reduce the costs of building hardware routers, over a potentially great number of nodes. We want to minimize the local memory requirement for a distributed routing strategy.

Table routing is a standard solution to the shortest path routing problem for arbitrary networks. At each node in the network is stored a table listing for each possible destination the output port that should be used to send a message to that node along a shortest path. That solution guarantees shortest paths but requires $\Theta(n \log \Delta)$ bits of space per node, where n is the number of nodes and Δ is the maximum degree of a node.

To alleviate the space requirements of routing tables, *compact routing schemes* were introduced: in [SK85] for arbitrary networks and in [FJ88, FJ89, FJ90] for planar and c-decomposable networks. Trade-offs between the space requirements for every node and the length of the routes were proposed

*This is a revised version of the report titled *Worst Case Bounds for Shortest Path Interval Routing* written in January 1995. This revised report present an improvement on the section 5.

in [ABNLP90, AP92, PU88]. A popular compact routing method, *interval routing*, is to group together the destination nodes corresponding to the same output port of a given node in intervals. Just as for table routing, this method requires that a header of only $O(\log n)$ bits be added to the forwarded message. This routing scheme was introduced in [SK85], generalized in [vLT87] and shortest path interval routing was discussed in [BvLT91, FJ88, vLT87].

Let us model a network of processors as a connected, simple and loop-less symmetric digraph $G = (V, E)$, where V denotes the set of vertices of G (corresponding to the routers) and E the set of arcs of G (corresponding to the set of directed links of the symmetric network). We assume that the cost of sending a message along any arc of G is uniform. An *interval* $[a, b]$ of the set $\{1, \ldots, n\}$ is the set of consecutive integers $\{a, \ldots, b\}$ cyclically. For example, $[7, 2]$ is the subset $\{7, 8, 1, 2\}$ of $\{1, \ldots, 8\}$.

Given a symmetric digraph $G = (V, E)$ of n vertices, an *interval routing scheme* $R = (\mathcal{L}, \mathcal{I})$ for G consists of:

1. a one-to-one labeling function of the vertices, $\mathcal{L} : V \to \{1, \ldots, n\}$
2. a family $\mathcal{I} = \{I_e, e \in E\}$, where I_e is a set of intervals of $\{1, \ldots, n\}$ associated with arc e

Moreover $\mathcal{L}$ and $\mathcal{I}$ must be such that the following properties hold:

i. for every $x \in V$, $\mathcal{L}(x) \cup \{I_{(x,y)} | (x, y) \in E\} = \{1, \ldots, n\}$
(we know how to route messages from x to every node in G)

ii. for every two distinct arcs (x, y) and (x, z) of E, $I_{(x,y)} \cap I_{(x,z)} = \emptyset$
(the routing scheme is well-defined)

We say that a routing R is a *shortest path* routing scheme if the node-to-node routes induced by R always use a shortest path in G. From now on, we will only consider shortest path interval routing schemes.

If such an interval scheme R is defined on a graph G, then message routing is performed as follows: upon reception of a message, vertex x first compares the message header, $\mathcal{L}(y)$ with its own label, $\mathcal{L}(x)$, to check if the message has arrived at its destination. If not, then the message and its header are forwarded through the unique arc (x, z) such that $\mathcal{L}(y) \in I_{(x,z)}$.

Consider Figure 1 as an example of a shortest path interval routing scheme. In this example, the labeling function $\mathcal{L}$ maps vertices a, b, e, g, d, f, c to integers $1, \ldots, 7$ respectively. In this graph, which is shown with undirected edges, the set of intervals $I_{(x,y)}$ assigned to arc (x, y) is placed close to vertex x. For example, the intervals $[1, 2]$ and $[5]$ that are close to vertex 7 correspond to the set of integers $\{1, 2, 5\}$, and form the set $I_{(7,1)}$ that labels the arc $(7, 1)$. Accordingly, if the vertex b wants to send a message to vertex f under R, the message will follow arcs (b, a), (a, c) and (c, f) in order. Note that in this model, the label of the node may or may not belong to an interval of one of its outgoing arcs (e.g. for this graph, the node labeled 5 does, while the node labeled 3 does not).

Routing strategies that do not require shortest paths have been studied in [BvLT91, FG94a, SK85], where the authors give a complete characterization of the graphs requiring a small number

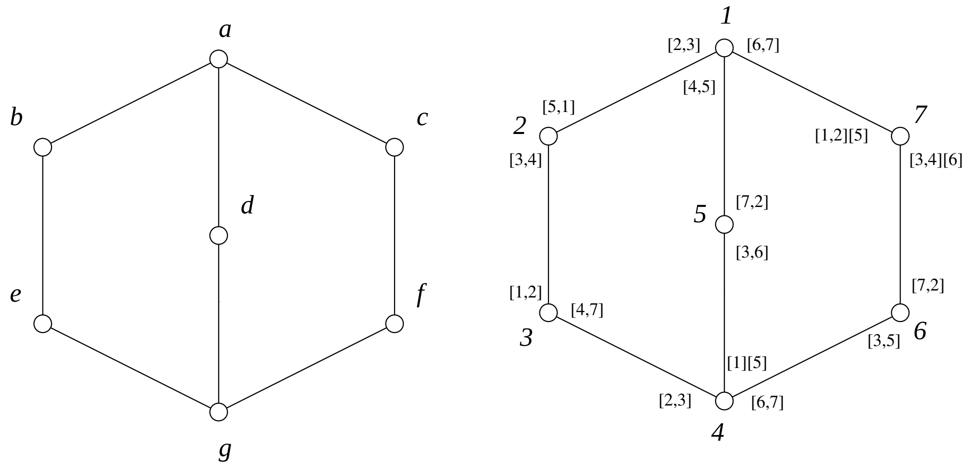


Figure 1: A graph G and an interval routing scheme for G.

of intervals for different restricted versions of interval routing. A hardware solution to the routing problem based on intervals was proposed by INMOS with its C104 chip (see [DFL93] and [MTW93]).

Given a graph G of n vertices and an interval routing scheme R for G, we define $\mathrm{IRS}(R)$ to be the maximum over all the arcs of G of the number of intervals that is required to encode the destinations associated to that arc ($\mathrm{IRS}(R) = \max_e |I_e|$ such that $I_e \in \mathcal{I}$). We define the *compactness of a graph* G, denoted $\mathrm{IRS}(G)$, as $\min_R \mathrm{IRS}(R)$ for all shortest path interval routing schemes R on G. In a sense, $\mathrm{IRS}(G)$ is the maximum number of intervals required by the "most compact" shortest path interval routing scheme on G. Note that we consider in the following that graphs have at least one arc and thus the compactness of a graph is always greater than or equal to 1. Since interval routing was introduced in the hope of reducing the amount of space required, $\mathrm{IRS}(G)$ is an important parameter to consider. In fact, given a graph G, there is a node of G that requires $\Omega(\mathrm{IRS}(G)\log n)$ bits of local memory under a shortest path interval routing scheme.

Most of the work in the literature on shortest path interval routing has been concerned with finding $\mathrm{IRS}(G)$ for specific networks: chordal rings in [FGS94a], trees, hypercubes, d-meshes, d-tori and r-complete-bipartite graphs in [BvLT91, FG94c, KKR93], unit-interval and unit-circular networks in [FG94c]. It is shown in [FG94b, FvLS94, KKR93, Ruš88] that $\mathrm{IRS}(G)$ is not bounded by a constant in the general case. In this paper we are interested in finding a worst case graph G with a large compactness. For every integer n we define $\mathrm{IRS}(n) = \max_G \mathrm{IRS}(G)$ such that G has n vertices. $\mathrm{IRS}(n)$ is the maximum compactness for graphs of n vertices.

It was shown in [KKR93] that $\mathrm{IRS}(n) \in \Omega(\sqrt[3]{n})$. The result was then improved in [FvLS94], where it was shown that $\mathrm{IRS}(n) \in \Omega(n/\log n)$. In this paper, we present a general technique for proving lower bounds on $\mathrm{IRS}(n)$ and for every n, we exhibit a graph G of n nodes for which we can prove that $\mathrm{IRS}(G) \in \Theta(n)$. We then extend the techniques introduced to construct for every fixed Δ and every n a graph G_Δ of maximum degree $\Delta \geq 3$ and of n nodes for which $\mathrm{IRS}(G_\Delta) \in \Omega(n/(\log n)^2)$.

More precisely, if we let $n(k)$ denote the number of nodes of the smallest network G for which $\mathrm{IRS}(G) \geq k$, we show that $2k+1 \leq n(k) \leq 12k - 11$ for every integer $k \geq 2$, and thus that $\mathrm{IRS}(n) \geq n/12$. The lower bound of $2k+1$ on $n(k)$ was obtained in [FG94a] with the following simple argument: by the pigeon hole principle, any integer labeling on $n-1$ nodes can give at most

$\lfloor (n-1)/2 \rfloor$ wrap around intervals of consecutive integers. This lower bound on $n(k)$ proves that our bound on $\text{IRS}(n)$ is asymptotically tight. Now let $\text{IRS}(n, \Delta)$ denote the largest compactness of a graph of n vertices and of maximum degree Δ. We adapt the construction to show that $\text{IRS}(n, \Delta)$ is greater than $\frac{n+4\log_2 n+5}{4\log_2 n(3\log_2 n+1)}$, for sufficiently large n and for every $\Delta \geq 3$.

In next section, we introduce the matrices of constraints, which provide a general tool for proving lower bounds on $\text{IRS}(n)$ and on $\text{IRS}(n, \Delta)$. In the same section, we describe how to construct a graph of $p + 2q$ vertices from a $p \times q$ boolean matrix such that if this matrix requires k blocks of consecutive ones in one of its columns, then the constructed graph G satisfies $\text{IRS}(G) \geq k$. In section 3, we present results from coding theory that we apply to produce suitable matrices that we use together in section 4 with our construction of section 2.2 to establish a lower bound on $\text{IRS}(n)$. In section 5, we adapt the construction of section 2.2 to obtain a graph G_Δ of at most $6pq-4p-4q$ vertices and of maximum degree Δ, from any $p \times q$ boolean matrix. Then we use suitable matrices to establish a lower bound on $\text{IRS}(n, \Delta)$, for any n and any $\Delta \geq 3$.

2 Matrices, codes and graphs of constraints

2.1 Matrices and codes of constraints

Given an arbitrary connected graph G, computing $\text{IRS}(G)$ is generally difficult. In fact, the problem has been shown to be NP-hard in [FGS94b]. There seems to be no other way than checking the minimum number of intervals required by each shortest path interval routing scheme on G. In this section, we introduce a tool that is helpful in establishing lower bounds on $\text{IRS}(G)$. More precisely, this tool is a mean of reducing the problem of finding the compactness of a graph G to a problem on boolean matrices. This tool is based on the notion of a *matrix of constraints*, a concept that we now introduce.

Consider vertex b in the graph G drawn on the left hand side of Figure 1. Note first that the shortest path from vertex b to vertices a, c, d, e and g is unique. The shortest path from b to a, c and d must use arc (b, a), and the shortest path from b to e and g must use arc (b, e). There is a shortest path from b to f that uses arc (b, a), and another that uses arc (b, e). Either path may be used, and this choice depends on the routing scheme.

In general, for every triple of vertices (u, v, w) of a graph G where u and v are adjacent vertices and $u \neq w$, three cases may occur for shortest paths:

1. Every shortest path from u to w must use arc (u, v).
2. Every shortest path from u to w must not use arc (u, v).
3. There are shortest paths from u to w that use arc (u, v) and there are shortest paths from u to w that do not use arc (u, v).

For the first two cases, arc (u, v) forms a *constraint* for the vertex w on the graph G. Note that (u, v) is not a constraint for vertex u, since there is no shortest path from u to u. From a routing point of view, any scheme can, in a first step, checks if $u = w$ and thus no adjacent arc of u must be used. It is the case for shortest path interval routing scheme, where the label of u may or may not belong to intervals associated to each adjacent arcs of u.

A *matrix of constraints* of a symmetric digraph $G = (V, E)$ is a $p \times q$ boolean matrix $M = (m_{i,j})$ whose rows are labeled with vertices of a subset $\{v_1, \ldots, v_p\}$ of V and whose columns are labeled with arcs of a subset $\{e_1, \ldots, e_q\}$ of E, such that:

1. $m_{i,j} = 1$ if and only if every shortest path from the tail of e_j to vertex v_i uses arc e_j.
2. $m_{i,j} = 0$ if and only if no shortest path from the tail of e_j to vertex v_i uses arc e_j.

Consider a column (u, v) of a $p \times q$ matrix of constraints and suppose that the vertices of the graph have been labeled with integers by a shortest path interval routing scheme $R = (\mathcal{L}, \mathcal{I})$. If there is a 1 at the intersection of the column with the row labeled by vertex w, then the label of w must be on arc (u, v) in R, i.e. $\mathcal{L}(w) \in I_{(u,v)}$. Similarly, if there is a 0 then the label of w cannot be on arc (u, v) in R, i.e. $\mathcal{L}(w) \notin I_{(u,v)}$. If we permute the rows of the matrix such that the integer labels of the rows are placed in ascending order in the matrix, then clearly the number of blocks of consecutive 1's in column (u, v) is a lower bound on the number of intervals for that arc in R. Table 1 shows a matrix of constraints for the graph of Figure 1.

$$M = \begin{array}{cc} \begin{array}{ccccccc} (c,a) & (e,b) & (g,d) & (c,f) & (e,g) & (g,e) & (g,f) \end{array} & \\ \boxed{\begin{array}{ccccccc} 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}} & \begin{array}{c} b \\ a \\ d \\ f \end{array} \end{array}$$

Table 1: A matrix of constraints for the graph of Figure 1.

Note that a matrix obtained by permuting the rows of a matrix of constraints is also a matrix of constraints for the same graph and corresponds to a relabeling of the vertices (the integer labels have to be in ascending order). If we can show that under any permutation of the rows of the matrix there must be at least one column with a certain number k of blocks of consecutive 1's, then the graph must require at least k intervals. Finding a matrix of constraints for a graph G and establishing a bound on the maximum number of blocks of consecutive 1's in a column, minimized over all permutations of the rows of the matrix, therefore yield a lower bound on $\textsc{Irs}(G)$. We now formalize these ideas.

A *(p,q)-code* is a non empty family C of $p \times q$ boolean matrices such that *(i)* if M is in C then any other matrix M' of C can be obtained by permuting the rows and columns of M and *(ii)* all the matrices obtained by permuting the rows and columns of M are in C. A (p, q)-code can be seen as an equivalence class of the set of $p \times q$ boolean matrices, using row and column permutation as a congruence operator. It therefore makes sense to specify a code C with a representative matrix from C.

Given a boolean matrix M, let $\mathrm{I}(M)$ denote the *compactness of* M, that is the maximum, over all columns of M, of the number of blocks of consecutive 1's cyclically (the first and the last bit in the boolean word formed by a column of M are considered consecutive). For example, in the boolean matrix of Figure 2, the first and second columns have one blocks of consecutive 1's, while the third has two. Therefore $\mathrm{I}(M') = 2$ for this matrix, while $\mathrm{I}(M) = 1$ for the matrix of constraints of table 1. Note that $\mathrm{I}(M) = 0$ if and only if $M = (0)$.

We extend our definition of compactness to codes by defining the *compactness of a* (p, q)*-code* C, denoted $\mathrm{I}(C)$ to be the minimum of $\mathrm{I}(M)$ over all matrices M in C. If M is a matrix of constraints

$$
M' = \begin{array}{c} (a_1, b_1) \quad (a_2, b_2) \quad (a_3, b_3) \\ \boxed{\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{array}} \end{array} \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array}
$$

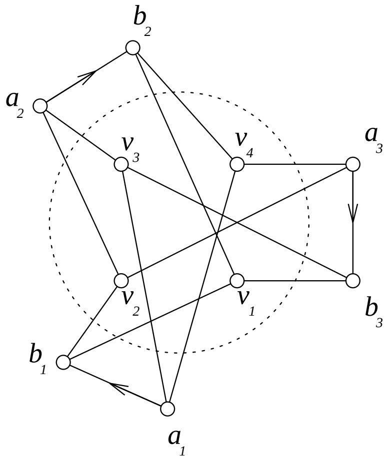


Figure 2: A graph for a $(4, 3)$-code of compactness 2.

of a graph G, then we call the family of matrices obtained by permuting the rows and columns of M a *(p,q)-code of constraints* of G. We are now ready to introduce the main result of this section.

Lemma 1 ([FvLS94]) *If C is a code of constraints of graph G then,* $\textsc{Irs}(G) \geq \mathrm{I}(C)$.

Proof. Let R be any shortest path interval routing for a graph G and let C be any code of constraints of G. Consider the set of integers used by R to label the vertices that label the rows of every matrix in the code. Let M be a matrix of C for which the integers, that now label the rows, are in ascending order from top to bottom. (Note that any matrix of C obtained by permuting only the columns of M satisfies this condition). Since there is a column of M with at least $\mathrm{I}(C)$ blocks of consecutive 1's by definition, it follows that the arc corresponding to the column has at least $\mathrm{I}(C)$ intervals under routing scheme R (there are at least $\mathrm{I}(C)$ "holes" when we list the integers corresponding to the arc). Since R is an arbitrary routing scheme for G, it follows that $\textsc{Irs}(G) \geq \mathrm{I}(C)$. □

Unfortunately we do not know of a better relation between the compactness of a graph and the compactness of the "less compact" code of constraints of the graph. The graph drawn on Figure 3 (page 11) is an example where there exists a code of constraints with the same compactness of the graph itself (it was already proved in [FG94c] that the compactness of this graph is 2). But Proposition 1 of appendix B establishes that in general, there is no code of constraints with the same compactness as the graphs'.

Also, it is not necessarily easy to compute $\mathrm{I}(C)$ in general. Indeed, in [FGS93] it is proved that given a (p, q)-code C, computing $\mathrm{I}(C)$ is NP-hard. This result is derived from the CONSECUTIVE ONES SUBMATRIX NP-complete problem in [GJ77]. However, it can be decided with a polynomial time algorithm if $\mathrm{I}(C) = 1$ (see [BL76]). Anyhow, the result of Lemma 1 is useful for constructing graphs for which we want to guarantee a given number of intervals.

This idea of matrix of constraints was independently introduced in [FvLS94] where the authors deal with the concept of *unique matrix* of shortest path representation. With this concept, they construct a graph of n vertices that requires $\Omega(n/\log n)$ intervals on one specific arc. In the following section we extend this concept to the idea of *graphs of constraints*, a more powerful tool to improve their lower bound.

2.2 Graphs of constraints

We present below how to construct a symmetric digraph $G = (V, E)$ from a $p \times q$ boolean matrix $M = (m_{i,j})$ such that M is a matrix of constraints of G. To simplify presentation, we describe G as being undirected. Refer to Figure 2 for an example of the construction using a 3×4 matrix. We get:

Lemma 2 *For every $p \times q$ boolean matrix M, there exists a graph G of $p + 2q$ vertices such that M is a matrix of constraints of G.*

Proof. Let $M = (m_{i,j})$ be any $p \times q$ boolean matrix. We construct a graph G composed of two layers. The bottom layer is a set of p independent vertices, $\{v_1, \ldots, v_p\}$, and the top layer consists of q copies of K_2 (the complete graph of two vertices). We denote a_j and b_j the two vertices of the jth copy of K_2, for $1 \leq j \leq q$. Hence the set of vertices of G is $\{v_1, ..., v_p, a_1, b_1, ..., a_q, b_q\}$; G has $p + 2q$ vertices.

We connect the vertices belonging to different layers as follows. If $m_{i,j} = 1$ then add the edge[1] $\langle b_j, v_i \rangle$ and if $m_{i,j} = 0$ then add the edge $\langle a_j, v_i \rangle$. For each of the q edges $\langle a_j, b_j \rangle$, we add exactly p edges and so G has $pq + q$ edges. It is clear that G thus constructed is connected and has a diameter less than 3.

We prove that the graph G that we constructed from a boolean matrix M has M as matrix of constraints. We first construct a $p \times q$ matrix of constraints M' of G as follows: label column j of M' with arc (a_j, b_j), for $1 \leq j \leq q$, and label row i of M' with vertex v_i, for $1 \leq i \leq p$. By construction M' is a matrix of constraints of G and clearly $M' = M$. Therefore M is a matrix of constraints of the graph G. $\square$

Remark. Let $\Delta(G)$ denote the maximum degree of graph G. Let M be a $p \times q$ boolean matrix, and G its graph of constraints built as in Lemma 2. It is easy to see that $\Delta(G) \geq \max\{z/q, p - z/q\}$, where z is the total number of 0 entries in the matrix M (consider any edge $\langle a_i, b_i \rangle$ of the construction).

The graph built in the proof of Lemma 2 from a boolean matrix M is called a *graph of constraints of the matrix M*. Since a permutation of the rows and columns of a matrix of constraints of a graph can be seen as a relabeling of the vertices and arcs respectively, then for a code C and for any two matrices M and M' in C, if G is a graph of constraints of matrix M then it is also a graph of constraints of matrix M'. Thus we can speak of a *graph of constraints of a code C*.

3 Codes with large compactness

In this section we present a method for constructing (p, q)-codes C with a large value of $\mathrm{I}(C)$ as a function of (p, q). We first extend our definition of code as follows. An (p, q, d)*-code* C is a

[1]To avoid a confusion with intervals, we denote $\langle a, b \rangle$ the edge connecting vertices a and b.

(p,q)-code such that for every matrix M in C, every two rows of M differ in at least d places (d may be 0). For example the well-known Gray codes of length p are the $(2^p, p, 1)$-codes. We use $A(q,d)$ to denote the largest value of p for which there is a (p,q,d)-code. Note that, in general the exact value of $A(q,d)$ is unknown, but tables can be found in the literature (e.g. [MS77]). The following Lemma is useful in finding a lower bound on $\mathrm{I}(C)$ for a given code C.

Lemma 3 ([FvLS94]) *For every (p,q,d)-code C, $\mathrm{I}(C) \geq pd/(2q)$.*

Proof. Consider a boolean matrix M of the (p,q)-code C. For $i \in \{1,\ldots,p\}$, let d_i be the Hamming distance between the two consecutive rows, i and $i+1$ (modulo p, the last and the first row are consecutive) of M, and for $j \in \{1,\ldots,q\}$ let k_j denote the number of blocks of consecutive 1's in column j of M. We call $D = \sum_{i=1}^{p} d_i$ the total Hamming distance over all rows of M and $K = \sum_{j=1}^{q} k_j$ the total number of blocks of consecutive 1's in M. It is easy to see that each raising of a block of consecutive 1's provides an increment of two on the total distance D, i.e $2K = D$. Since $d_i \geq d$ for every i, we have $2\sum_{j=1}^{q} k_j = \sum_{i=1}^{p} d_i \geq pd$. By the pigeon hole principle, since M has q columns, there must be a column j, $1 \leq j \leq q$, such that $k_j \geq pd/(2q)$. $\Box$

The fact that the Hamming distance yields a lower bounds on the number of intervals was due to [Fla94].

Corollary 1 *For every integer $k \geq 2$, if there exist integers p, q and d such that $A(q,d) \geq p > 2q(k-1)/d$, then there exists a (p,q)-code C such that $\mathrm{I}(C) \geq k$.*

Proof. Assume that for some integer k, $k \geq 2$, there exists a triple (p,q,d) such that $A(q,d) \geq p > 2q(k-1)/d$. Since $p \leq A(q,d)$, then there exists, by definition of $A(q,d)$, a (p,q,d)-code C. Applying Lemma 3, $\mathrm{I}(C) \geq pd/(2q) \geq k$. $\Box$

Therefore, to find a code with large compactness using Lemma 3, we may use a (p,q,d)-code maximizing $pd/(2q)$. With the next lemma we will see that there exists a (p,q)-code C with $q \in \Theta(p)$ and $\mathrm{I}(C) \in \Theta(p)$. We use the well-known Hadamard code for that purpose.

Lemma 4 *For every integer $\delta \geq 1$, there exists a $(2^{\delta+2}, 2^{\delta+1}, 2^{\delta})$-code.*

Proof. We show here how to construct a representative matrix M_δ of a $(2^{\delta+2}, 2^{\delta+1}, 2^{\delta})$-code, by induction. The basis, when δ is 1, is the following $(8,4,2)$-code

$$M_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Suppose, as our inductive hypothesis, that there exists a $(2^{\delta+2}, 2^{\delta+1}, 2^{\delta})$ Hadamard code C_δ. Then let M_δ be a representative matrix of C_δ and let $M_{\delta+1}$ as follows:

$$M_{\delta+1} = \begin{bmatrix} M_\delta & M_\delta \\ M_\delta & \overline{M_\delta} \end{bmatrix}$$

where $\overline{M_\delta}$ is the matrix M_δ with every bit complemented. Notice that if every two rows of M_δ differ in exactly 2^δ bits, except for the pair of rows $(000\ldots0)$ and $(111\ldots1)$, i.e. the first and third row of M_δ, then it is easy to see that this fact will also hold for $M_{\delta+1}$. Since it is invariant under permutation of rows and columns, we can conclude that the family $C_{\delta+1}$ of matrices generated from $M_{\delta+1}$, is indeed a $(2^{\delta+3}, 2^{\delta+2}, 2^{\delta+1})$-code. □

The $(2^{\delta+2}, 2^{\delta+1}, 2^{\delta})$-code constructed in the proof of Lemma 4 will be denoted C_δ from now on. Lemma 3, we know that $\mathrm{I}(C_\delta) \geq 2^\delta$. In appendix A, we will prove that in fact, $\mathrm{I}(C_\delta) = 2^\delta$.

Actually, MacWilliams and Sloane in [MS77] proved that a $(8\delta, 4\delta, 2\delta)$-code exists if there exists an *Hadamard matrix* of dimensions $8\delta \times 4\delta$. According to them, Hadamard matrices were known in 1977 for every δ less than 70. Lemma 4 is in fact a corollary of the existence of Sylvester matrices.

4 A lower bound for Irs(n)

Lemma 5 *Let $n(k)$ be the minimum order of a graph G such that* $\textsc{Irs}(G) \geq k$. *Then for all $k \geq 2$, $n(k) \leq 2^{\lceil \log_2 k \rceil + 2} + 4k - 3$.*

Proof. Let $k \geq 2$. By Corollary 1 there exists an (p, q, d)-code C with $\mathrm{I}(C) \geq k$ if $A(q, d) \geq p > 2q(k-1)/d$. Let M be a $p \times q$ boolean matrix of such a (p, q, d)-code C. Lemma 2 guarantees the existence of a graph of constraints G of the matrix M and, applying Lemma 1, such that $\textsc{Irs}(G) \geq \mathrm{I}(C) \geq k$. Hence the number of vertices of G is an upper bound on $n(k)$, i.e. $n(k) \leq \min_{p,q} |V(G)|$, if $\mathrm{I}(C) \geq k$. Therefore

$$\forall k \geq 2, n(k) \leq \min_{p,q,d} (p + 2q) \text{ where } (p, q, d) \text{ satisfies } A(q, d) \geq p > 2q(k-1)/d$$

Using the $(2^{\delta+2}, 2^{\delta+1}, 2^{\delta})$-code C_δ of Lemma 4 with $\delta = \lceil \log_2 k \rceil$, $q = 2^{\delta+1}$ and $d = 2^\delta$, we obtain $A(q, d) \geq p > 2q(k-1)/d$ if and only if $2^{\delta+2} \geq p > 4(k-1)$. Indeed, Lemma 4 shows that $A(2^{\delta+1}, 2^\delta) \geq 2^{\delta+2}$ and, by applying Plotkin's bound [MS77], which states that $A(2i, i) \leq 4i$ for i even, we get that $A(2^{\delta+1}, 2^\delta) = 2^{\delta+2}$. As p should be the smallest integer such that $2^{\delta+2} \geq p > 4(k-1)$, we can choose $p = 4k - 3$ (we can remove three rows at least of C_δ) to yield the desired result. □

Lemma 5 is enough to prove that $n(k) \in \Theta(k)$ for all $k \geq 2$, because we have shown $2k+1 \leq n(k)$. The following theorem is a direct consequence of Lemma 5.

Theorem 1 *For every integer $n \geq 2$,* $\textsc{Irs}(n) \geq n/12$.

Proof. Since for any $k \geq 2$, $2^{\lceil \log_2 k \rceil} \leq 2(k-1)$, from Lemma 5 we get that $n(k) \leq 12k - 11$. By definition of $n(k)$, we derive that for every integer $n \geq 2$ there exists a graph G with n vertices such that $\textsc{Irs}(G) \geq k \geq \lfloor (n+11)/12 \rfloor$. Therefore $\textsc{Irs}(G) \geq n/12$. □

Remark. We can see that the graph G built in Lemma 5 has an unbounded maximum degree. G was obtained using the $(2^{\delta+2}, 2^{\delta+1})$-code C_δ, which contains the same number of 0 and 1 entries. Thus, applying the remark of section 2.2, we get $\Delta(G) \geq 2^\delta + 1/2$. Indeed we remove some rows of C_δ, but at least $2^{\delta+2}/2 + 1$ rows of C_δ are left, and thus at least $2^{\delta+1}(2^{\delta+2}/2+1)/2$ 0 or 1 entries are left. Since the number of vertices of G is $n = 2^{\delta+2} + 4k - 3$ and k is such that $2^{\delta-1} < k \leq 2^\delta$, then $\Delta(G) \geq n/8$.

We now show that it is possible to tighten the upper bound on $n(k)$. For k a power of 2, we obtained $n(k) \leq 8k - 3$ in Lemma 5. If we consider the case where $k = 2$ for example, the upper bound on $n(2)$ of Lemma 5 is obtained by using a $(5,4,2)$-code: $d = 2^{\lceil \log_2 2 \rceil} = 2$, $q = 2d = 4$ and $p = 4k - 3 = 5$. Thus the bound of Lemma 5 states that $n(2) \leq p + 2q = 5 + 2 \cdot 4 = 13$. But if instead we would use the $(4,3,2)$-code of Figure 2, then we would find, by applying Lemma 3, that $n(2) \leq p + 2q = 4 + 2 \cdot 3 = 10$, an improvement. As another example, consider the case $k = 5$. The bound of Lemma 5 gives $n(5) \leq 49$ by using a $(17,6,3)$-code. But we can use a $(25,6,2)$-code instead (the readers are welcome to convince themselves that such a code exists) to find $n(5) \leq p + 2q = 25 + 2 \cdot 6 = 37$. Therefore, Lemma 5 does not provide an optimal result and it can be improved upon.

We can compute a smaller upper bound on $n(k)$ using the construction of a graph of constraints of Lemma 2, based on the fact that $n(k) \leq \min_{p,q,d}(p + 2q)$ such that $A(q,d) \geq p > 2q(k-1)/d$. Using a table of the best lower bounds known on $A(q,d)$ taken from [MS77], we find the following upper bounds on $n(k)$, for $k \leq 21$ (see table 2), by applying the above minimization on a computer with a systematic search. This table also gives the values (p,q,d) of the (p,q,d)-code C used for the construction of a graph of constraints of code C. The graph corresponding to the first row of this table, $k = 2$ using an $(4,3,2)$-code and given $n(2) \leq 10$, is shown in Figure 2.

k	d	q	p	Upper Bound $n(k)$
2	2	3	4	**10**
3	2	5	11	**21**
4	2	5	16	**26**
5	2	6	25	**37**
6	2	6	31	**43**
7	4	10	31	**51**
8	4	10	36	**56**
9	6	14	38	**66**
10	6	14	43	**71**
11	6	14	47	**75**

k	d	q	p	Upper Bound $n(k)$
12	6	14	52	**80**
13	6	14	57	**85**
14	6	14	61	**89**
15	8	18	64	**100**
16	6	15	76	**106**
17	6	15	81	**111**
18	6	15	86	**116**
19	6	15	91	**121**
20	6	15	96	**126**
21	6	15	101	**131**

Table 2: Upper bound on $n(k)$ for small values of k.

Remark. We showed that, by using techniques from coding theory, we were able to obtain asymptotically tight bounds on the size of the smallest graph which requires at least k intervals to route along shortest paths using interval routing. Nevertheless, we believe that our bound can be improved upon. For example, consider once more the case $k = 2$. Applying the general result of

Lemma 5, we showed that $n(2) \leq 10$. However it was shown by a case analysis that there exist a graph of seven vertices (see section 2.1 and Figure 3) of compactness 2, and therefore that $n(2) \leq 7$. In appendix C we prove that in fact $n(2) = 7$. Though asymptotically tight, there still exist a small gap between our upper bounds and the exact values of $n(k)$.

$$C = \begin{array}{ccc|c} (b,g) & (d,g) & (f,g) & \\ \hline 1 & 1 & 1 & g \\ 1 & 0 & 0 & e \\ 0 & 1 & 0 & a \\ 0 & 0 & 1 & c \end{array}$$

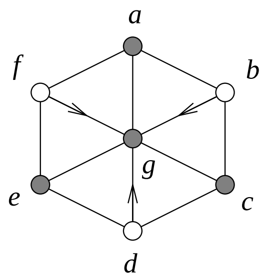


Figure 3: A code of constraints C for a graph G with $\textsc{Irs}(G) = \mathrm{I}(C) = 2$.

5 Worst case graphs of bounded degree

We have seen that our lower bound on $\textsc{Irs}(n)$, in Lemma 5, is achieved with a graph of constraints of order n with a maximum degree in $\Theta(n)$. In this section, we establish a lower bound on $\textsc{Irs}(n, \Delta)$ for shortest path interval routing schemes on graphs of order n and of maximum degree Δ. We will prove that $\textsc{Irs}(n, \Delta) \in \Omega(n/(\log n)^2)$, for every integer n and for every integer $\Delta \geq 3$. We assume, in the following, that $\Delta \geq 3$ because graphs with maximum degree less than 3 clearly have a compactness of 1 (see [SK85] for Trees and [vLT87] for Rings). In this section, we will construct a graph of constraints G with maximum degree Δ from an arbitrary $p \times q$ boolean matrix M, such that M is a matrix of constraints of G. We will then prove our bound on $\textsc{Irs}(n, \Delta)$ by using the results of sections 2.1.

Refer to Figure 4 for an illustration of the construction. Assume that we are given a $p \times q$ boolean matrix M and an integer $\Delta \geq 3$. The symmetric digraph G, that we describe as undirected, is composed of three main levels of vertices denoted Low, Medium and High, and of intermediary vertices (drawn in black in Figure 4). The Low level is a set of p independent vertices that we denote v_i, $1 \leq i \leq p$; the High level consists of q copies of K_2, that we denote $\langle a_j, b_j \rangle$, $1 \leq j \leq q$; the Medium level is composed of pq vertices labeled $w_{i,j}$, $1 \leq i \leq p$ and $1 \leq j \leq q$.

These three levels are connected with trees of maximum degree Δ. Let us first describe how the Low and the Medium levels are connected. At every vertex v_i of the Low level, we root a tree T_{v_i} whose leaves are the $w_{i,j}$'s, for $j \in \{1, \ldots, q\}$. All the trees T_{v_i}'s are isomorphic. They are minimal undirected trees of maximum degree Δ with exactly q leaves. In each T_{v_i}'s, $i \in \{1, \ldots, p\}$, the leaves $w_{i,j}$, $j \in \{1, \ldots, q\}$, are all at a distance $h' = 1 + \lceil \log_{\Delta-1}(q/\Delta) \rceil$ from the root v_i. We denote by r the number of vertices in any tree T_{v_i}, $i \in \{1, \ldots, p\}$ (all trees have the same number of vertices).

To connect the Medium and High levels, we use a second type of tree. At vertex a_j (resp. b_j), $j \in \{1, \ldots, q\}$, we root a $(\Delta-1)$-ary tree T_{a_j} (resp. T_{b_j}) the root has maximum degree $\Delta - 1$ while the other vertices have maximum degree Δ. In T_{a_j} (resp. T_{b_j}), $j \in \{1, \ldots, q\}$, the leaves are the vertices $w_{i,j}$, for every i such that $m_{i,j} = 0$ (resp. $m_{i,j} = 1$). Furthermore, every leaf of the tree T_{a_j} (resp. T_{b_j}) is at distance $h = \lceil \log_{\Delta-1} \max_i(\max\{z_i, p - z_i\}) \rceil$ from a_j (resp. b_j), where z_i is the number of 0's in column i of the $p \times q$ matrix M. T_{a_j} and T_{b_j} have the smallest possible number of vertices necessary to satisfy these requirements. We denote by r_{a_j} and r_{b_j} respectively the number of vertices of trees T_{a_j} and T_{b_j}, $j \in \{1, \ldots, q\}$.

In fact, the vertices $w_{i,j}$ can be seen as a grid, where the $w_{i,j}$'s of row i are connected by tree T_{v_i}, while some of the $w_{i,j}$ of column j are connected by tree T_{a_j} and the others by tree T_{b_j}. The total number of vertices in graph G is equal to $\sum_{j=1}^{q}(r_{a_j} + r_{b_j}) + pr - pq$ (the $w_{i,j}$'s are counted twice).

Lemma 6 *For every $p \times q$ boolean matrix M and every integer $\Delta \geq 3$, there exists a graph of constraints of matrix M of maximum degree Δ and with $6pq - 4p - 4q$ vertices at most.*

Proof. Let us consider the preceding construction of the graph of constraints G of M. The trees T_{a_j} and T_{b_j} can be constructed as follows: starting from the leaves (at most p), construct a full $(\Delta-1)$-ary tree, adding intermediary nodes as required. From the root of that tree, construct a path to a_j (or b_j), so that T_{a_j} (and T_{b_j}) has height h. To built T_{v_i}, we start from the root v_i with Δ children (or q if $q \leq \Delta$) and then we root, in each child, q full $(\Delta-1)$-ary trees of height $h-1$. Then we can remove $\Delta(\Delta-1)^{h-1} - q$ leaves from the last stage. T_{v_i}, T_{a_j} and T_{b_j} therefore always exist, and so the above construction guarantees that we obtain a graph G that is connected and has a maximum degree Δ.

We now prove that M is a matrix of constraints of G. We will first construct a $p \times q$ matrix of constraints of G, M', and we will then show that M' and M are equal. Label the p rows of $M' = (m'_{i,j})$ with the p vertices of the Low level (the v_i's) and label the q columns of M' with the arcs (a_j, b_j)'s of the High level. For each $j \in \{1, \ldots, q\}$, let A_j (resp. B_j) be the set of v_i's such that $w_{i,j}$ is a leaf of the tree T_{a_j} (resp. T_{b_j}). Clearly trees T_{a_j} and T_{b_j} are disjoint and thus A_j and B_j partition the Low level vertices.

We now compute the entries of matrix M', according to the definition of matrix of constraints. We first show that if $v_i \in B_j$, then every shortest path from a_j to v_i must use the arc (a_j, b_j). Indeed, the path from a_j to v_i has a length of $h + h' + 1$ (go to b_j in one step, take h steps down T_{b_j} to reach $w_{i,j}$, and then h' steps up T_{v_i} to v_i). If we assume, for the sake of contradiction, that the shortest path between a_j and v_i leaves through an arc of T_{a_j}, then the length of the path must be at least: h using tree T_{a_j} to reach a vertex $w_{i',j}$, then 2 at least using tree $T_{v'_i}$ to reach a vertex $w_{i',j'}$, then 2 at least using tree $T_{a_{j'}}$ or $T_{b_{j'}}$ to reach vertex $w_{i,j'}$, and finally h', using tree T_{v_i} to reach the vertex v_i. The path would therefore at least be of length $h+h'+4$, greater than $h+h'+1$. Hence $m'_{i,j}$ is 1.

Now, suppose that vertex $v_i \in A_j$. We will show that the shortest path from a_j to v_i must use an arc of T_{a_j}, i.e. it must not use arc (a_j, b_j). First, it is simple to see that there is a path from a_j to v_i of length $h+h'$: take h steps down T_{a_j} to vertex $w_{i,j}$, and then h' steps up T_{v_i} to v_i. Following an argument similar to the one given above (about the path of length $h + h' + 4$) one can see that a path from a_j to v_i that uses arc (a_j, b_j) must have a length of $1 + h + 2 + 2 + h'$ at least. Hence $m'_{i,j}$ is 0. Therefore M' is a matrix of constraints of G since all its entries are well-defined. It is now easy to see that $M' = M$: if $v_i \in B_j$ (and $m'_{i,j} = 1$), then by definition of the set B_j, vertex $w_{i,j}$ is a leaf of T_{b_j}. But by construction of G, this is possible only if $m_{i,j} = 1$. Similarly, if $v_i \in A_j$ (and $m'_{i,j} = 0$) then $m_{i,j} = 0$.

Let us now compute the number of vertices of G. We first consider T_{v_i}, the smallest tree with

q leaves at equal distance from the root, with maximum degree Δ (T_{v_i} differs from a $(\Delta-1)$-ary tree in that its root can have degree Δ). This tree can have at most $\Delta(\Delta-1)^{i-1}$ nodes a level i away from the root, and thus $r \leq 1 + \Delta \sum_{i=0}^{h'-2}(\Delta-1)^i + q$. By choosing $h' = 1 + \lceil \log_{\Delta-1}(q/\Delta)\rceil$ and $\Delta = 3$, then $r \leq 3 \cdot 2^{h'-1} + q - 2$. Note that to have a (p,q)-code with compactness $k \geq 2$ we must have $p \geq 4$ and $q \geq 3$. The reader can check the fact that no smaller boolean matrix has a compactness of $k \geq 2$. But since $3 \leq q \leq 3 \cdot 2^{\lceil \log_2(q/3)\rceil} \leq 2(q-1)$, it follows that $r \leq 3q-4$.

We now need to bound the value of $r_{a_j} + r_{b_j}$, $j \in \{1, \dots, p\}$. T_{a_j} is the smallest $(\Delta-1)$-ary tree of height $h = \lceil \log_{\Delta-1}(p)\rceil$ with z leaves, each at distance h from the root, where z is the maximum number of 0's in a column of M, over all columns of M ($z \leq p$). We already described T_{a_j} and T_{b_j} at the beginning of the proof. In a $(\Delta-1)$-ary tree there are at most $(\Delta-1)^i$ nodes a level i, and thus $p \leq (\Delta-1)^h$ (we remove some vertices to the last level if necessary). Let z_j be the number of 0's in column j of M. The height of the tree T_{a_j} is $h_{a_j} = \lceil \log_{\Delta-1}(z_j)\rceil$ or 1 if $z_j = 0$, and the height of tree T_{b_j} is $h_{b_j} = \lceil \log_{\Delta-1}(p - z_j)\rceil$ or 1 if $z_j = p$. Note that $h_{a_j} + h_{b_j} \geq 2$. Hence $r_{a_j} \leq \sum_{i=0}^{h_{a_j}-1}(\Delta-1)^i + z_j + h - h_{a_j}$, where $h - h_{a_j}$ is the length of the added path from the root of the full tree to a_j. Similarly, $r_{b_j} \leq \sum_{i=0}^{h_{b_j}-1}(\Delta-1)^i + p - z_j + h - h_{b_j}$. Set $\Delta = 3$. If $1 < z_j < p$, then we have $r_{a_j} + r_{b_j} \leq 2^{\lceil \log_2(z_j)\rceil} + 2^{\lceil \log_2(p-z_j)\rceil} - 2 + p + 2\lceil \log_2 p\rceil - 2$. And since $\log_2 p \leq p/2$, $r_{a_j} + r_{b_j} \leq 2(z_j - 1) + 2(p - z_j - 1) + p - 4 + 2\lceil p/2\rceil$. Since $2\lceil p/2\rceil \leq p+1$, it follows that $r_{a_j} + r_{b_j} \leq 4p-7$. If $z_j \leq 1$ or if $z_j = p$, then $2^{h_{a_j}} + 2^{h_{b_j}} \leq 1 + 2(p-1)$, and thus $r_{a_j} + r_{b_j} \leq 4p-4$.

In any case, $r_{a_j} + r_{b_j} \leq 4(p-1)$ and $r \leq 3q-4$. Thus we obtain the desired result on the number of vertices of G. □

The following lemma gives a construction of (p,q)-codes for particular values of p and q.

Lemma 7 *For every integer $\delta \geq 2$, there exists a $(2^{\delta-1}, \delta, 2)$-code.*

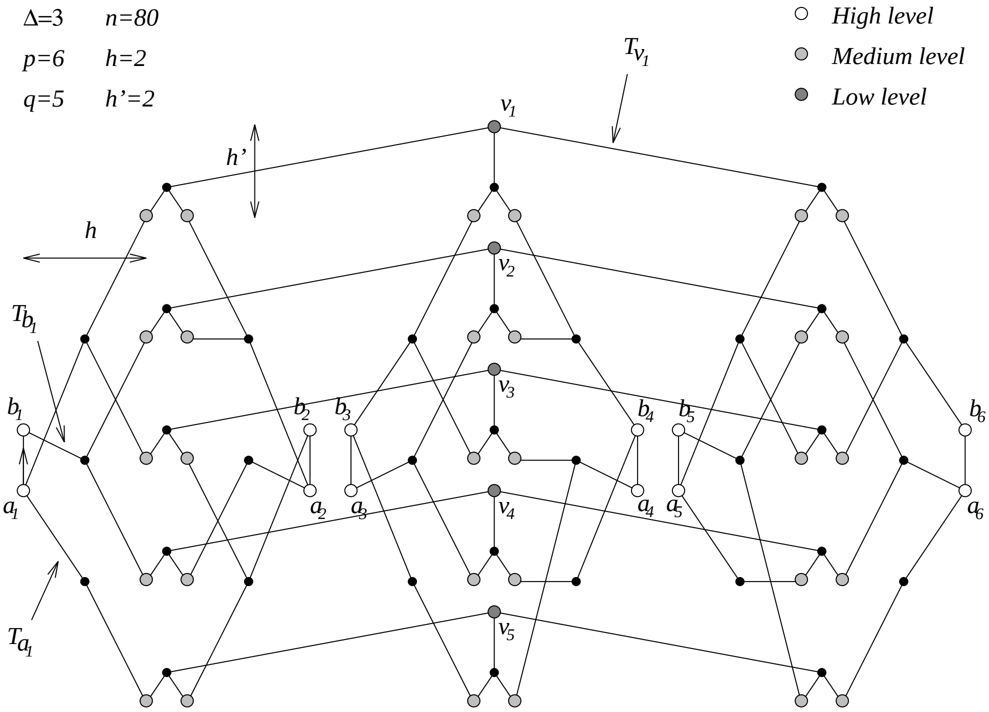
Proof. Let $P = (v_1, \dots, v_{2^\delta})$ be a Hamiltonian path in the Hypercube of dimension $\delta \geq 2$. Let C be the $(2^{\delta-1}, \delta)$-code such that row r_i of C, $i \in \{1, \dots, 2^{\delta-1}\}$, is the standard binary representation of the vertex v_{2i-1} of P. By construction, any two rows r_i and r_j of C, $i \neq j$, differ by at least 2 places, since two adjacent vertices in the Hypercube are at an odd distance in P. Thus C is a $(2^{\delta-1}, \delta, 2)$-code. □

Finally we derive the following theorem:

Theorem 2 *For any sufficiently large integer n ($n \geq 44$) and for every $\Delta \geq 3$,*

$$\textsc{Irs}(n, \Delta) \geq \frac{n + 4\log_2 n + 5}{4\log_2 n(3\log_2 n + 1)} \in \Omega(n/(\log n)^2)$$

Proof. Let n be a sufficiently large integer, let $p = 2^{q-1}$ and let q such that $6pq - 4p - 4q = 3q2^q - 2^{q+1} - 4q \leq n < 3(q+1)2^{q+1} - 2^{q+2} - 4(q+1)$. Applying Lemma 6, for every (p,q)-code C there exists a graph of constraints G of the code C of maximum degree Δ and with n' vertices, $n' \leq n$. We construct a graph G' with a maximum degree Δ from G with exactly n vertices by adding a path of $n - n'$ vertices connected to one of the $w_{i,j}$'s vertices of G, which are all of degree two. Clearly $\textsc{Irs}(G') \geq \textsc{Irs}(G)$, since G is a *subgraph of shortest paths* of G' (see [FG94c]). Hence,



$$M = \begin{array}{cccccc|c} (a_1,b_1) & (a_2,b_2) & (a_3,b_3) & (a_4,b_4) & (a_5,b_5) & (a_6,b_6) & \\ \hline 0 & 0 & 1 & 1 & 0 & 1 & v_1 \\ 1 & 0 & 0 & 1 & 1 & 0 & v_2 \\ 0 & 1 & 1 & 0 & 0 & 1 & v_3 \\ 1 & 0 & 0 & 1 & 0 & 0 & v_4 \\ 0 & 1 & 1 & 0 & 1 & 0 & v_5 \end{array}$$

Figure 4: Graph of constraints of the matrix M with $n = 80$ vertices and with a maximum degree $\Delta = 3$.

for every $\Delta \geq 3$, $\textsc{Irs}(n, \Delta) \geq \textsc{Irs}(G)$. Thus by applying Lemma 1, for every $(2^{q-1}, q)$-code C, $\textsc{Irs}(n, \Delta) \geq \mathrm{I}(C)$. Applying Lemma 7, let C be a $(2^{q-1}, q, 2)$-code for $q \geq 2$. Therefore, applying Lemma 3, $\mathrm{I}(M) \geq \mathrm{I}(C) \geq 2^{q-1}/q$.

$$n < 3(q+1)2^{q+1} - 2^{q+2} - 4(q+1) \quad \Rightarrow \quad n + 4q + 5 \leq \frac{2^{q-1}}{q} 4q(3q+1) \quad \Rightarrow \quad \frac{n+5}{4q(3q+1)} + \frac{1}{3q+1} \leq \mathrm{I}(C)$$

By assumption, $n \geq 3q2^q - 2^{q+1} - 4q \geq 2^q$, for $q \geq 3$ and $n \geq 44$. Therefore $\log_2 n \geq q$ and finally,

$$\textsc{Irs}(n, \Delta) \geq \frac{n+5}{4\log_2 n(3\log_2 n + 1)} + \frac{1}{3\log_2 n + 1} \in \Omega(n/(\log n)^2)$$

□

The same techniques as in section 4 can be used to improve the general upper bound for $n_\Delta(k)$, the number of vertices of the smallest graph G_Δ with maximum degree Δ for which $\textsc{Irs}(G_\Delta) \geq k$.

We would have to compute $\min_{p,q} |V(G)|$ such that $\mathrm{I}(C) \geq k$ and such that G is a graph of constraints of (p,q)-code C with maximum degree Δ.

6 Conclusion

From a local memory requirement point of view, we have seen that for a graph G of n vertices, the minimum number of intervals required to perform shortest path interval routing on G, $\textsc{Irs}(G)$, is an important parameter to consider, since at least one router of G needs to store $\Omega(\textsc{Irs}(G)\log n)$ bits of information. By proving upper bounds on $n(k)$, the smallest number of vertices of a graph of compactness greater than k, we showed that there exist a worst case graph that requires a router to have $\Theta(n\log n)$ bits of local memory. Therefore, interval routing schemes are not better than routing tables in the general case of unbounded degree graphs. However for bounded maximum degree graphs, our worst case still uses only $\Omega(n/\log n)$ bits locally, compared to $O(n)$ bits for routing tables. It would be interesting to determine whether or not there is a graph G_Δ of n vertices and of maximum degree Δ such that $\textsc{Irs}(G_\Delta) \in \Omega(n/\log n)$. If no such graph exists, then the class of bounded degree graphs is a large class of graphs for which interval routing schemes are better than tables.

A Compactness of Hadamard codes

The lower bound on $\text{I}\textsc{rs}(n)$ is based on the lower bound on $\text{I}(C_\delta)$, where C_δ is defined in the proof of Lemma 4. We showed in Lemma 4 that $\text{I}(C_\delta) \geq 2^\delta$. We now show, by proving an upper bound on $\text{I}(C_\delta)$, that the lower bound on $\text{I}\textsc{rs}(n)$ cannot be improved using this code.

Theorem 3 *For any $\delta \geq 1$, $\text{I}(C_\delta) = 2^\delta$.*

Proof. Since we already know that $I(C_\delta) \geq 2^\delta$, we prove by induction on δ that there is a matrix M_δ of the $(2^{\delta+2}, 2^{\delta+1}, 2^\delta)$-code C_δ such that $\text{I}(M_\delta) = 2^\delta$. As our base case, we use the matrix M_1 that is the same matrix as in the proof of Lemma 4. A sequence of matrices M_δ, $\delta \geq 1$, is obtained as follows. Given M_δ, $\delta \geq 1$, partition the rows of M_δ in blocks of four consecutive rows starting with the top of the code, and number these groups in order, starting at 0. We say that a block is *even* (or *odd*) if its label is even (or odd respectively). We obtain matrix $M_{\delta+1}$ from matrix M_δ with the following three steps:

1. (Shuffle step) In each block of four vectors, exchange the position of the two middle rows. This step groups together the rows that are complements. For example, using the matrix M_1 above, the shuffle step yields the following matrix:

$$\left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right]$$

2. (Doubling step) Replace every row (A) of M_δ with the row (AA). For example, replace the row (0011) above with (00110011).

3. (Extend step) After the doubling step, groups of two consecutive rows consist of a binary vector (AA) and its complement $(\overline{AA})$. Now, insert row $(A\overline{A})$ after row (AA), and insert row $(\overline{A}A)$ after row $(\overline{AA})$. For example, the block of two rows

$$\left[\begin{array}{cccccccc} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right]$$

becomes the following block of four rows:

$$\left[\begin{array}{cccccccc} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

It is easy to show by induction that for every $\delta \geq 1$, M_δ has the property that rows 1 and 3 and rows 2 and 4 of every group of four rows are the complement of each other. It is also easy to see that the matrix $M_{\delta+1}$ thus obtained is the same as the one of Lemma 4, with its rows permuted i.e., $M_{\delta+1} \in C_{\delta+1}$. We are therefore only left with proving that $\mathrm{I}(M_\delta) = 2^\delta$.

We now consider how the procedure above transforms a given column of M_δ (note that the doubling step has no effect here). Without loss of generality, consider only one of the 2^δ leftmost columns of the matrix. We assume, as our inductive hypothesis, that the columns of M_δ satisfy the following two properties:

1. In every column, the even groups of four bits have the pattern (0011) or (1100). in every column, the odd groups have the pattern (0110) or (1001).

2. Every column has exactly 2^δ blocks of consecutive 1's (cyclically).

These properties can easily be seen to hold for matrix M_1. We have already seen that after the shuffling step, the bits of every block of two bits (rows $2i$ and $2i+1$) in a column (of M_δ, now being transformed) differ. Therefore, after the extend step of the construction, the blocks of four bits in a column of $M_{\delta+1}$ (obtained from the blocks of two bits after the shuffling step) with an even label will have the patterns (0011) (from (01)) or (1100) (from (10)) since for these we only insert a copy of each bit. For the blocks of four with an odd label, the extend step of our construction inserts after each bit the complement of that bit. We therefore obtain the patterns (0110) (from (01)) or (1001) (from (10)). Property 1 therefore holds for $M_{\delta+1}$.

We can now prove that property 2 above holds for $M_{\delta+1}$. We first show that the shuffle step adds $2^{\delta-1}$ to the total number of intervals of each column, and then we show that the extend step also adds $2^{\delta-1}$ to the total number of intervals of each column. Each column of the matrix $M_{\delta+1}$ therefore has $2^{\delta-1} + 2^{\delta-1} + 2^\delta$ (inductive hypothesis) $= 2^{\delta+1}$ intervals.

Suppose again, without loss of generality, that we consider one of the 2^δ leftmost columns of M_δ. Partition the column in blocks of eight bits, starting from the top. The first four bits of each block corresponds to an even block of four. By property 1, there are only four possible patterns for the blocks of eight bits:

a. (0011 0110)
b. (0011 1001)
c. (1100 0110)
d. (1100 1001)

After the shuffle step, each of the four patterns gets transformed to:

a'. (0101 0110)
b'. (0101 1001)
c'. (1010 0110)
d'. (1010 1001)

In each case, exactly one interval gets added to the column since the boundaries of every block remain unchanged. The shuffle step therefore adds exactly $2^{\delta+2}/8 = 2^{\delta-1}$ intervals to each column in total.

Now, re-partition the column obtained from the shuffle step into blocks of eight bits such that the first four bits of each block correspond to an odd block of four. We obtain the following four patterns:

A. (0110 0101)
B. (1001 0101)
C. (0110 1010)
D. (1001 1010)

After the extend step, these patterns get transformed to the following (e.g. pattern A gets transformed to pattern A'):

A'. (0110 1100 0110 0011)
B'. (1001 0011 0110 0011)
C'. (0110 1100 1001 1100)
D'. (1001 0011 1001 1100)

Each of A', B', C' and D' has the same boundaries as A, B, C and D respectively, and since each has one interval more than its counterpart, we can conclude that the extend step together add $2^{\delta+3}/16 = 2^{\delta-1}$ new intervals to each column. The results extend to the last 2^{δ} columns of M_{δ} by changing the partitions. Therefore, property 2 holds for $M_{\delta+1}$, which conclude the proof. $\square$

B Compactness of codes and graphs

Proposition 1 *There exists a graph G such that* $\textsc{Irs}(G) > \mathrm{I}(C)$ *for every code of constraints C of G.*

Proof. Consider the graph G of 7 vertices drawn in Figure 1. We have to construct all the codes of constraints of G and check that each code has a compactness of at most 1. In fact it is enough to check for (p,q)-codes with $p \geq 4$ and $q \geq 3$ since it is easy to see that any smaller code has a compactness of at most 1. Furthermore, we only need to check for $p \leq 6$ because, we assumed that any arc (u,v) is not a constraint for the vertex u. Therefore the number of rows p of any code of constraint of G is at most 6. We leave it to the reader to check that every (p,q)-code of constraints C of G with $4 \leq p \leq 6$ and $3 \leq q$ has a compactness of at most one, i.e. $\mathrm{I}(C) \leq 1$.

We now prove that $\textsc{Irs}(G) = 2$. Assume that there exists a shortest path interval routing scheme on G, $R = (\mathcal{L}, \mathcal{I})$, with $\textsc{Irs}(R) = 1$. To simplify the presentation of the proof, we set $x = \mathcal{L}(x)$, for $x \in \{a,b,c,d,e,f,g\}$. Also, if X and Y are two subsets of vertices of G, we say that $X < Y$ if for every $(x,y) \in X \times Y$, $x < y$. Since vertices b and c are isomorphic, assume without loss of generality that $b < c$. The order $d < b < a < c$ (circular order modulo 7) is impossible because the interval assigned to arc (g, d), $\mathrm{I}_{(g,d)}$, must contain d and a but neither b nor c. Thus we have $\{a, d\} < b < c$. Arcs (d, a) and (d, g) establish the condition $\{e, f, g\} < \{a, b, c\}$. Therefore $\{a, e, f, g, d\} < b < c$. $I_{g,f}$ must contain f and c but neither d nor b, thus $f < d < b < c$. Moreover $I_{g,e}$ must contain e and b but neither d nor c, thus $d < e < b < c$. And finally we have $f < d < e < b < c$. Now we must have $f < \{d, g\} < e < b < c$, since $I_{a,d}$ must contain d and g but neither f nor e nor b nor c. Similarly we must have $f < \{a, d, g\} < e < b < c$, because $I_{g,d}$ must contain d and a but neither f nor e nor b nor c. But this last condition is incompatible with the condition $\{e, f, g\} < \{a, b, c\}$. This contradiction shows that $\textsc{Irs}(G) > 1$. Figure 1 gives a shortest path interval routing scheme R with $\textsc{Irs}(R) = 2$, and therefore $\textsc{Irs}(G) = 2$. $\Box$

C Smallest graphs of compactness 2

In this appendix we prove that the minimum graph of compactness 2 has 7 vertices, i.e. $n(2) = 7$. For this, we use the list of all graphs of order less than 7, which can be found in [Har69]. The following lemmas will reduce the number of cases to consider. In the following, all graphs are described as symmetric digraphs.

Lemma 8 *Let G be a 1-vertex-connected graph. The compactness of G is the maximum of the compactness overall subgraph of G composed of one 2-vertex-connected component of G and of its neighbor cutvertices in G.*

Proof. Let $G = (V, E)$ be a 1-vertex-connected graph. A subgraph of G composed of one 2-vertex-connected component of G and of its neighbor cutvertices in G, is denoted a *2-component* of G. Let A be a 2-component of G. Clearly A is a *subgraph of shortest paths* [FG94c] of G, i.e. a subgraph that contains all the shortest paths between any pair of vertices of A. Let $k = \max_S \textsc{Irs}(S)$, for any 2-component S of G. Therefore, applying Theorem 2 of [FG94c], we get that $\textsc{Irs}(G) \geq k \geq \textsc{Irs}(A)$.

We now prove that $\textsc{Irs}(G) \leq k$. The proof is constructive: 1) decompose G in 2-components, 2) successively merge these 2-components and their shortest path interval routing scheme to obtain an shortest path interval routing scheme on G. Phase 2) merge two 2-components at the first step. It results a subgraph of G that is no more a 2-component of G. In fact, in the remaining merging, we merge subgraphs that are non 2-component of G but that have a cutvertex in common. Let us show how to do a merging in general.

Since G is a 1-vertex-connected graph, there exists a cutvertex x of G and we can decomposed G in two subgraphs, $A = (V_A, E_A)$ and $B = (V_B, E_B)$, such that $V_A \cup V_B = V$ and $V_A \cap V_B = \{x\}$. We assume by induction that A and B are of compactness at most k, and we will prove that $\textsc{Irs}(G) \leq k$.

Let $n_A = |V_A|$ and $n_B = |V_B|$. Let $R_A = (\mathcal{I}_A, \mathcal{L}_A)$ and $R_B = (\mathcal{I}_B, \mathcal{L}_B)$ be two shortest path interval routing schemes on graphs A and B respectively, such that $\mathcal{L}_A(x) = n_A$ and $\mathcal{L}_B(x) = 1$. Conditions $\mathcal{L}_A(x) = n_A$ and $\mathcal{L}_B(x) = 1$ are not restrictive, since clearly every circular permutation composed with the labeling function defines an interval routing scheme with same compactness and isomorphic set of routing paths.

We define a shortest interval routing scheme $R = (\mathcal{L}, \mathcal{I})$ on G as follows: $\mathcal{L}(v) = \mathcal{L}_A(v)$ for all vertices v of graph A, and $\mathcal{L}(w) = \mathcal{L}_B(w) + n_A - 1$, for all vertices $w \neq x$ of graph B. We shift also any single interval $[a, b] \in \mathcal{I}_B$ to get a new set $\mathcal{I}'_B$ of intervals of the form $[a + n_A - 1, b + n_A - 1]$. We extend sets $\mathcal{I}_A$ and $\mathcal{I}'_B$ to obtain a set of intervals $\mathcal{I}$ for G as follows : for any single interval $I = [a, b]$ of $\mathcal{I}_A$ or of $\mathcal{I}'_B$ containing the integer n_A, let $I' = I \cup [n_A, n_A + n_B - 1]$. I' is composed of only one interval since all its elements are consecutive. We finally set $\mathcal{I}$ as the union of extended intervals sets of $\mathcal{I}_A$ and $\mathcal{I}'_B$. It is easy to see that the shortest path defined by R between two vertices of the same subgraphs A or B are the same as in R_A or R_B, and any shortest path between a vertex u of A and a vertex w of B, must travel the cutvertex x, which belongs to set of vertices of A and of B.

The compactness of R is less than $\max\{\text{IRS}(A), \text{IRS}(B)\}$, therefore $\text{IRS}(G) = k$. $\square$

The following lemma will be useful to check quickly if a graph with many edges has compactness 1.

Lemma 9 *Let G be a connected graph of n vertices having d vertices of degree $n-1$. Let m be the number of edge-connected components having at least two vertices in the complement graph of G. If $d \geq (n-m)/2$ then the compactness of G is 1.*

Proof. Let G be a connected graph of n vertices. Assume that G has d vertices of degree $n-1$. Let $\overline{G}$ denote the complement graph of G. $\overline{G}$ is composed of m connected components $A_1, \ldots, A_m$ of order at least two and of d single vertices. Assume that $d \geq (n-m)/2$. Without loss of generality, we assume that $m \geq 1$, since otherwise G is simply the complete graph. For each connected component A_i, $i \in \{1, \ldots, m\}$, we root a spanning tree T_i at any vertex of A_i. Let n_i denote the number of vertices of A_i.

We now construct a shortest path interval routing scheme $R = (\mathcal{L}, \mathcal{I})$ on G. For $i \in \{1, \ldots, m\}$ and for every vertex x of A_i, we set $\mathcal{L}(x) = 2j - 1 + \sum_{k=1}^{i-1} n_k$ in a depth first search scheme according to T_i, for all $j \in \{1, \ldots, n_i\}$. Since $n - d \geq 2m$, $m \geq 1$ and $d \geq (n-m)/2$, then $d \geq n - d - m \geq 1$, and thus we can label $n - d - m$ of the d single vertices y of $\overline{G}$ with $\mathcal{L}(y) = 2j$, for all $j \in \{1, \ldots, n-d-m\}$. For the other single vertices y' of $\overline{G}$, if they exist, we set $\mathcal{L}(y') = k$, for all $k \in \{2(n-d-m)+2, \ldots, n\}$. The set $\mathcal{I}$ is defined as follows:

i. For the d vertices x of G of degree $n-1$, we assign $I_{(x,y)} = [\mathcal{L}(y)]$ for the $n-1$ vertices y's connected to x.

ii. For vertices of G with a degree strictly lower than $n-1$, we assign intervals as follows: let $i \in \{1, \ldots, m\}$ and x be a vertex of A_i. For every arc (x, y) of G, we assign the interval $I_{(x,y)} = [\mathcal{L}(y) - 1, \mathcal{L}(y)]$ if x and y are not adjacent in G, and $I_{(x,y)} = [\mathcal{L}(y)]$ otherwise.

Hence all vertices of G are labeled and the compactness of R is 1.

To prove that R is connected and define a shortest path routing scheme, consider any two vertices x and y of G. Since $n > d \geq 1$, i.e. G has at least one vertex of degree $n-1$ and $G \neq K_n$, then G has diameter 2. Hence either x and y are adjacent or there is a third vertex z of G such that z is connected to both x and y. If x and y are adjacent, then we have $\mathcal{L}(y) \in I_{(x,y)}$ in both cases *(i)* and *(ii)*, and by symmetry $\mathcal{L}(x) \in I_{(y,x)}$. Otherwise x and y are not adjacent in

G and thus, there exists an $i \in \{1, \ldots, m\}$ such that x and y together belong to A_i. Therefore $\mathcal{L}(y) \in I_{(x,y)} = [\mathcal{L}(z) = \mathcal{L}(y) - 1, \mathcal{L}(y)]$, by *(ii)* since neither x nor y are of degree $n-1$. And by symmetry we also get that $\mathcal{L}(x) \in I_{(y,x)}$. We have thus proved thus that all paths built by R are shortest paths. □

Theorem 4 *$n(2) = 7$.*

Proof. We have already showed that $n(2) \leq 7$ (Figure 1 and refer to Proposition 1). Moreover $n(2) \geq 5$ since $n(k) \geq 2k+1$ for any $k \geq 2$. We will check for every graphs of 5 and 6 vertices that they admit a shortest path interval routing scheme of compactness 1. Since outerplanar graphs have a compactness of 1 [FJ88], we need not check them. Since any connected graph of 4 vertices or less has a compactness of 1, then, applying Lemma 8 and Lemma 9, only 2 graphs of 5 vertices must be checked: $K_{2,3}$ and the graph composed of a cycle of 4 vertices with $K_{1,3}$ connected by its three vertices. Referring to the representation of these graphs pages 217 of [Har69], a circular labeling and a straightforward assignment of intervals give a shortest path interval routing scheme of compactness 1. Hence $n(2) \geq 6$. Similarly, since every connected graph of 5 vertices has a compactness of 1, then, applying Lemma 8 and Lemma 9, only 43 graphs of 6 vertices, on pages 220–224 of [Har69], must be checked. A circular labeling and a straightforward assignment of intervals give a shortest path interval routing scheme of compactness 1 for all these graphs, except for the one, composed of a cycle of 4 vertices and a path of length 3 connecting two non adjacent vertices. For this graph, we can label the vertices $(1, 2, 6, 5, 4, 3)$ given the circular representation of this graph on page 220 [Har69]. Therefore $n(2) \geq 7$. □

Acknowledgments

Thanks to Milan Mosny for a helpful discussion regarding the result of appendix A. We are very thankful to Pierre Fraigniaud for thoroughly reading this manuscript and for his incisive comments and helpful suggestions.

References

[ABNLP90] Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Improved routing strategies with succint tables. *Journal of Algorithms*, 11:307–341, February 1990.

[AP92] Baruch Awerbuch and David Peleg. Routing with polynomial communication-space trade-off. *SIAM Journal on Discrete Math*, 5(2):151–162, May 1992.

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs and graphs planarity using *pq*-tree algorithms. *Journal of Computer and System Science*, 13:335–379, 1976.

[BvLT91] Erwin M. Bakker, Jan van Leeuwen, and Richard B. Tan. Linear interval routing. *Algorithms Review*, 2:45–61, 1991.

[DFL93] Frédéric Desprez, Eric Fleury, and Michel Loi. T9000 et C104 : La nouvelle génération de transputers. Technical Report 93-01, LIP-ENS Lyon, LIP, École Normale Supérieure de Lyon, 69364 Lyon Cedex 07, France, February 1993.

[FG94a] Pierre Fraigniaud and Cyril Gavoille. A characterisation of networks supporting linear interval routing. In ACM PRESS, editor, *13th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 216–224, August 1994.

[FG94b] Pierre Fraigniaud and Cyril 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, January 1994.

[FG94c] Pierre Fraigniaud and Cyril Gavoille. Optimal interval routing. In Bruno Buchberger and Jens Volkert, editors, *Parallel Processing: CONPAR '94 - VAPP VI*, volume 854 of Lecture Notes in Computer Science, pages 785–796. Springer-Verlag, September 1994.

[FGS93] Michele Flammini, Giorgio Gambosi, and Sandro Salomone. Boolean routing. In Springer Verlag, editor, *7th Int. Workshop on Distributed Algorithms (WDAG)*, volume 725 Lecture Notes in Computer Science, pages 219–233, 1993.

[FGS94a] Michele Flammini, Giorgio Gambosi, and Sandro Salomone. Interval labeling schemes for chordal rings. In Carleton University School of Computer Science, editor, *Colloquium on Structural Information and Communication Complexity (SICC)*, Ottawa, Canada K1S 5B6, May 1994.

[FGS94b] Michele Flammini, Giorgio Gambosi, and Sandro Salomone. Interval routing schemes. (to appear in STACS'95), November 1994.

[FJ88] Greg N. Frederickson and Ravi Janardan. Designing networks with compact routing tables. *Algorithmica*, pages 171–190, 1988.

[FJ89] Greg N. Frederickson and Ravi Janardan. Efficient message routing in planar networks. *SIAM Journal on Computing*, 18(4):843–857, August 1989.

[FJ90] Greg N. Frederickson and Ravi Janardan. Space-efficient message routing in c-decomposable networks. *SIAM Journal on Computing*, 19(1):164–181, February 1990.

[Fla94] Michele Flammini. Private communications in New York, June 1994.

[FvLS94] Michele Flammini, Jan van Leeuwen, and A. Marchetti Spaccamela. Lower bounds on interval routing. Technical Report 69, Università di L'Aquila, Dipartemento di matematica Pura ed Applicata, October 1994.

[GJ77] Michael R. Garey and David S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1977.

[Har69] Frank Harary. *Graph Theory*. Addison-Wesley, 1969.

[KKR93] Evangelos Kranakis, Danny Krizanc, and S. S. Ravi. On multi-label linear interval routing schemes. In *19th International Workshop on Graph - Theoretic Concepts in Computer Science - Distributed Algorithms (WG)*, volume 790 of Lecture Notes in Computer Science, pages 338–349, Utrecht, June 1993. Springer-Verlag.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The Theory of Error-Correcting Codes.* North Holland, 1977.

[MTW93] M.D. May, P.W. Thompson, and P.H. Welch. Networks, routers and transputers: Function, perfomance, and applications. Technical report, inmos, SGS-THOMSON, 1993.

[PU88] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 43–52, Chicago, IL, May 1988.

[Ruš88] Peter Rušička. On efficient of interval routing algorithms. In M.P. Chytil, L. Janiga, and V. Koubek, editors, *Mathematical Foundations of Computer Science (MFSC)*, volume 324 of Lectures Notes in Computer Science, pages 492–500. Springer-Verlag, 1988.

[SK85] Nicola Santoro and Ramez Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, 1985.

[vLT87] Jan van Leeuwen and Richard B. Tan. Interval routing. *The Computer Journal*, 30(4):298–307, 1987.