(ℓ, k)-**Routing on Plane Grids**

Omid Amini, Florian Huc, Janez Zerovnik

▶ **To cite this version:**

HAL Id: inria-00265297

https://hal.inria.fr/inria-00265297v2

Submitted on 25 Mar 2008

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

(ℓ, k)-*Routage dans les Grilles Planaires*

Omid Amini — Florian Huc — Ignasi Sau — Janez Žerovnik

N° 6480

February 2008

Thème COM

*Rapport
de recherche*

(ℓ, k)-Routage dans les Grilles Planaires [*]

Omid Amini [†] , Florian Huc [‡] , Ignasi Sau [‡ §] , Janez Žerovnik [¶ ‖]

Thème COM — Systèmes communicants
Projet MASCOTTE

Résumé : Le problème de routage de paquets joue un rôle très important dans les réseaux de télécommunication. Il consiste à envoyer des données entre les sommets d'un réseau en un temps raisonable. Dans le (ℓ, k)-routage, chaque sommet peut envoyer au plus ℓ paquets et en recevoir k. Le routage de permutation correspond au cas $\ell = k = 1$. Dans le routage r-central, tous les noeuds à distance au plus r d'un sommet fixé envoient un paquet à ce sommet. Dans cet article nous étudions les problèmes précédents dans les grilles planaires. On utilise le modèle *store-and-forward* et Δ-port. Nous considérons les réseaux half et full-duplex. Les résultats principaux sont :

1. un algorithme *tight* pour le problème de routage de permutation dans les grilles hexagonales full-duplex, et dans les grilles hexagonales et triangulaires half-duplex.
2. un algorithme *tight* pour le routage r-central dans les grilles triangulaires et hexagonales.
3. un algorithme *tight* pour le (ℓ, ℓ)-routage dans les grilles carrées, triangulaires et hexagonales.
4. de bon algorithme d'approximation (en terme de temps de calcul) pour le (ℓ, k)-routage dans les grilles carrées, triangulaires et hexagonales. On donne aussi de nouvelles bornes inférieures sur le temps d'éxécution d'un algorithme qui utilise le routage par le plus court chemin.

Tous ces algorithmes sont complètements distribués.

Mots-clés : Routage de paquets, algorithme distribué, (ℓ, k)-routage, grilles planaires, routage de permutation, plus court chemin, algorithme sans mémoire

(ℓ, k)-Routing on Plane Grids **

Abstract: The packet routing problem plays an essential role in communication networks. It involves how to transfer data from some origins to some destinations within a reasonable amount of time. In the (ℓ, k)-routing problem, each node can send at most ℓ packets and receive at most k packets. Permutation routing is the particular case $\ell = k = 1$. In the r-central routing problem, all nodes at distance at most r from a fixed node v want to send a packet to v.

In this article we study the permutation routing, the r-central routing and the general (ℓ, k)-routing problems on plane grids, that is square grids, triangular grids and hexagonal grids. We use the *store-and-forward* Δ-port model, and we consider both full and half-duplex networks. The main contributions are the following:

1. Tight permutation routing algorithms on full-duplex hexagonal grids, and half duplex triangular and hexagonal grids.
2. Tight r-central routing algorithms on triangular and hexagonal grids.
3. Tight (k, k)-routing algorithms on square, triangular and hexagonal grids.
4. Good approximation algorithms (in terms of running time) for (ℓ, k)-routing on square, triangular and hexagonal grids, together with new lower bounds on the running time of any algorithm using shortest path routing.

All these algorithms are completely distributed, i.e. can be implemented independently at each node. Finally, we also formulate the (ℓ, k)-routing problem as a WEIGHTED EDGE COLORING problem on bipartite graphs.

Key-words: Packet routing, distributed algorithm, (ℓ, k)-routing, plane grids, permutation routing, shortest path, oblivious algorithm

1   Introduction

In telecommunication networks, it is essential to be able to route communications as quickly as possible. In this context, the *packet routing* problem plays a capital role. In this problem we are given a network and a set of packets to be routed through the nodes and the edges of the network graph. A packet is characterized by an origin and a destination node, and typically an edge can be used by no more than one packet at the same time. The objective is to find an algorithm to compute a schedule to route all packets which minimizes the total delivery time. This problem has been widely studied in the literature under many different assumptions. In 1988, Leighton, Maggs and Rao proved in their seminal article [31, 29] the existence of a schedule for routing any set of packets with edge-simple paths on a general network, in optimal time of $\mathcal{O}(C + D)$ steps. Here C is the congestion (maximum number of paths sharing an edge) and D the dilation (length of a longest path) and it is assumed that the paths are given a priori. The proof of [29] used Lovász Local Lemma and was non constructive. This result was further improved in [28] where the same authors gave an explicit algorithm, using the Beck's constructive version of the Local Lemma.

These algorithms to compute the optimal schedule are centralized. Then in [38] Ostrovsky and Rabani gave a distributed randomized algorithm running in $\mathcal{O}(C + D + \log^{1+\epsilon}(n))$ steps. We give a more detailed overview in Section 1.1.

Although these results are asymptotically tight, they deal with a general network, and in many cases it is possible to design more efficient algorithms by looking at specific packet configurations or network topologies. For instance, is it natural to bound the maximum number of messages that a node can send or receive. We focus on this point in Section 1.2, where we will formally define the problem under study in this paper.

On the other hand, the network under study plays a major role on the quality and the simplicity of the solution. For example, in a radio wireless environment, cellular networks are usually modeled by a *hexagonal grid* where nodes represent base stations. The cells of the hexagonal grids have good diameter to area ratio and still have a simple structure. If centers of neighboring cells are connected, the resulting graph is called a *triangular grid*. Notice that hexagonal grids are subgraphs of the triangular grid. We will talk about such networks in Section 1.3. In this paper we focus on the study of the (ℓ, k)-routing problem in convex subgraphs, i.e. that contain all shortest paths between all pairs of nodes of the square, triangular and hexagonal grid.

1.1   General Results on Packet Routing

In this section we provide a fast overview of the state-of-the-art of the general packet routing problem, in both the off-line and on-line settings in Sections 1.1.2 and 1.1.3 respectively, focusing mostly on the latter. We begin by recalling three classical lower bounds for the packet routing problem.

1.1.1   Classical lower bounds

In the packet routing problem, there are three classical types of lower bounds for the running time of any algorithm :

1. **Distance bound** : the longest distance over the paths of all packets (usually called *dilation*, denoted D) constitutes a lower bound on the number of steps required to route all the packets.
2. **Congestion bound** : the *congestion* of an edge of the network is defined as the number of paths using this edge. Then, the greatest congestion over all the edges of the network (denoted C) is also a lower bound on the number of steps, since at each step an edge can be used by at most one packet.
3. **Bisection bound** : Let $G = (V, E)$ be the graph which models the network, and $F \subseteq E$ a cut-set disconnecting G into two components G_1 and G_2. Let m be the number of packets with origin in G_1 and destination in G_2. Then, the number of routing steps used by any algorithm will be at least $\left\lceil \frac{m}{|F|} \right\rceil$.

1.1.2   Off-line routing

Given a set of packets to be sent through a network, a *path system* is defined as the union of the paths that each packet must follow. For a general network and any set of n demands, we have seen in Section 1.1.1 that the dilation and the congestion provide two lower bounds for the routing time. This proves that the *dilation* + *congestion* of a paths system used for the routing procedure is a lower bound of twice the routing time. In a celebrated paper, Leighton, Maggs and Rao proved the following theorem :

Theorem 1.1 ([31]) *For any set of requests and a path system for these requests, there is an* **off-line** *routing protocol that needs* $\mathcal{O}(C + D)$ *steps to route all the requests, where C is the congestion and D is the dilation of the path system.*

In addition, in [49] the authors show that, given the set of packets to be sent, it is possible to find in polynomial time a path system with $C + D$ within a factor 4 of the optimum. Thus, Theorem 1.1 can be announced in a more general way :

Theorem 1.2 ([49]) *For any set of requests, there is an* **off-line** *routing protocol that needs* $\mathcal{O}(C+D)$ *steps to route all the requests, where $C+D$ is the minimum congestion* + *dilation over all the possible path systems.*

Furthermore, this routing protocol uses fixed buffer size, i.e. the queue size at all nodes is bounded by a constant at each step. Nevertheless, it is important to notice that a huge constant may be hidden inside the $\mathcal{O}$ notation. As we have said in the introduction, this result was further improved in [28] where the same authors gave an explicit algorithm. These algorithms to compute the optimal schedule are centralized. In a distributed algorithm nodes must make their decisions independently, based on the packets they see, without the use of a centralized scheduler. Then in [38] Ostrovsky and Rabani gave a distributed randomized algorithm running in $\mathcal{O}(C + D + \log^{1+\epsilon}(n))$ steps.

We refer to Scheideler's thesis [45] for a complete compilation of general packet routing algorithms.

1.1.3 On-line routing

In the on-line setting, the oldest on-line protocol that deviates only by a factor logarithmic in n from the best possible runtime $\mathcal{O}(C + D)$ for arbitrary path-collections is the protocol presented by Leighton, Maggs and Rao in the same paper [31], running in $\mathcal{O}(C + D \log(Dn))$ steps with high probability. The authors call the algorithm on-line, rather than distributed. This schedule assumes that the paths are given a priori, hence it does not focus on the problem of choosing the paths to route the packets.

The results of [1] provide a routing algorithm that is $\log n$ competitive with respect to the congestion. In other words, it is worse than an optimal off-line algorithm only by a factor $\log n$. In this setting the demands arrive one by one and the algorithm routes calls based on the current congestion on the various links in the network, so this can be achieved only via centralized control and serializing the routing requests. In [3] the authors gave a distributed algorithm that repeatedly scans the network so as to choose the routes. This algorithm requires shared variables on the edges of the network and hence is hard to implement. Note that the two on-line algorithms above depend on the demands and are therefore *adaptive*. Recall that an *oblivious* routing strategy is specified by a path system $\mathcal{P}$ and a function w assigning a weight to every path in $\mathcal{P}$. This function w has the property that for every source-destination pair (s, t), the system of flow paths $\mathcal{P}_{s,t}$ for (s, t) fulfills $\sum_{q \in \mathcal{P}_{s,t}} w(q) = 1$. One can think of this function as a frequency distribution among several paths going from an origin s to a destination t. In *adaptive* routing, however, the path taken by a packet may also depend on other packets or events taking place in the network during its travel. Remark that every oblivious routing strategy is obviously on-line and distributed.

The first paper to perform a worst case theoretical analysis on oblivious routing is the paper of Valiant and Brebner [54], who considered routing on specific network topologies such as the hypercube. They give a randomized oblivious routing algorithm. Borodin and Hopcroft [6] and subsequently [22] have shown that deterministic oblivious routing algorithms cannot approximate well the minimal load on any non-trivial network.

In a recent paper, Räcke [40] gave the construction of a polylog competitive oblivious routing algorithm for general undirected networks. It seems truly surprising that one can come close to minimal congestion without any information on the current load in the network. This result has been improved in [4]. Lower bounds on the competitive ratio of oblivious routing have been studied for various types of networks. For example, for the d-dimensional mesh, Maggs et al. [35] gave the $\omega(\frac{C_*}{d}(\log n))$ lower bound on the competitive ratio of an oblivious algorithm on the mesh, where C_* is the optimal congestion.

So far, the oblivious algorithms studied in the literature have focused on minimizing the congestion while ignoring the dilation. In fact, the quality of the paths should be determined by the congestion C, and the dilation D. An open question is whether C and D can be controlled simultaneously. An appropriate parameter to capture how good is the dilation of a path system is the *stretch*, defined as the maximum over all packets of the ratio between

the length of the path taken by the routing protocol and the length of a shortest path from source to destination. In a recent work, Bush et al. [8] considered again the case of the d-dimensional mesh. They presented an on-line algorithm for which C and D are both within $\mathcal{O}(d^2)$ of the potential optimal, i.e. $D = \mathcal{O}(d^2 D_*)$ and $C = \mathcal{O}(dC_* \log(n))$, where D_* is the optimal dilation (remark that by [35], it is impossible to have a factor better than $\omega(\frac{C_*}{d}(\log n))$).

There is a simple counter-example network that shows that in general the two metrics (*dilation* and *congestion*) are orthogonal to each other : take an adjacent pair of nodes u, v and $\Theta(\sqrt{n})$ disjoint paths of length $\Theta(\sqrt{n})$ between u and v. For packets traveling from u to v, any routing algorithm that minimizes congestion has to use all the paths, however, in this way some packets follow long paths, giving high stretch. Nevertheless, in grids [8], and in some special kind of geometric networks [7] the congestion is within a poly-logarithmic factor from optimal and stretch is constant (d the dimension). As mentioned before an interesting open problem is to find other classes of networks where the congestion and stretch are minimized simultaneously [2]. Possible candidates for such networks could be for example bounded-growth networks, or networks whose nodes are uniformly distributed in closed polygons, which describe interesting cases of wireless networks.

The recent paper of Maggs [34] surveys a collection of theoretical results that relate the congestion and dilation of the paths taken by a set of packets in a network to the time required for their delivery.

1.2   Routing Problems

The initial and final positioning of the packets has a direct influence on the time needed for their routing. Considering static packet configuration, the most studied constraints refer to the maximum number of packets that a node can send and receive. Due to their practical importance, some of these problems have specific names :

1. **Permutation routing** : each node is the origin and the destination of at most one packet. To measure the routing capability of an interconnection network, the partial permutation routing (PPR) problem is usually used as the metric.
2. (ℓ, k)-**routing :** each node is the origin of at most ℓ packets and destination of at most k packets. Permutation routing corresponds to the case $\ell = k = 1$ of (ℓ, k)-routing. Another important particular case is the $(1, k)$-**routing**, in which each node can send at most one packet and receive at most k packets.
3. $(1, any)$-**routing** : each node is origin of at most one packet but there are no constraints on the number of packets that a node can receive.
4. r-**central routing** : all nodes at distance at most r of a central node send one message to this central node.

In all these problems we are given an initial packet configuration, and the objective is to route all packets to their respective destinations minimizing the total routing time, under the constraint that each edge can be used by at most one packet at the same time.

Besides of the constraints about the initial and final positions of the packets, there also exist different routing models at the intermediate nodes of the network. For instance, in the *hot potato model* no packet can be stored at the nodes of the network, whereas in the *store-and-forward* at each step a packet can either stay at a node or move to an adjacent node. Another widely used model is the *wormhole* routing.

On the other hand, one can consider constraints on the number of incident edges that each node of the network can use to send or receive packets at the same time. In the Δ-port model [16], each node can send or receive packets through all its incident edges at the same time.

In this article we study the store-and-forward Δ-port model. In addition, we suppose that cohabitation of multiple packets at the same node is allowed. I.e. a queue is required for each outgoing edge at each node.

The nature of the links of the network is another factor that influences the routing efficiency. The type of links is usually one of the following : full-duplex or half-duplex. In the full-duplex case there are two links between two adjacent nodes, one in each direction. Hence two packets can transit, one in each direction, simultaneously. In the half-duplex case only one packet can transit between two nodes, either in one direction of the edge or in the other. In this paper we focus on both half and full-duplex links.

1.3   Topologies

We now give a brief summary of various cases of (ℓ, k)-routing and $(1, any)$-routing that have been studied for several specific topologies. More precisely, in Section 1.3.1 we list the most important results for some networks which have attracted a great interest in the literature, like hypercubes and circulant graphs. Then we move to plane grids in Section 1.3.2. It is well known that there exist only three possible tessellations of the plane into regular polygons [55] : squares, triangles and hexagons. These graphs are those which we study in this article.

1.3.1   Different network topologies

In [20] the authors studied the permutation routing problem in low-dimensional hypercubes ($d \leq 12$). They gave optimal or good in the worst case oblivious algorithms, i.e. algorithms in which the path used by a packet is entirely determined by its origin and its destination. An other network widely studied in the literature is the two dimensional mesh with row and column buses. This network can also be diversified according to the capacities of the buses. In [51] Suel gave a deterministic algorithm to solve the permutation routing problem in such networks. It gives a schedule using at most $n + o(n)$ steps and a queue of size 2, where the queue is the maximum number of packets that have to be stored at an intermediate node. He also proposed a deterministic algorithm for r-dimensional arrays with buses working in $(2 - \frac{1}{r})n + o(n)$ steps and still using queues of size 2. In [27], the authors studied the (ℓ, ℓ)-routing problem in the mesh grid with two diagonals and gave, for $\ell \geq 9$ a deterministic algorithm using $\frac{2\ell n}{9} + (\ell n^{2/3})$ steps.

In [19], the authors introduced an algorithm called *big foot* algorithm. The idea of this algorithm is to identify two types of links and to move towards the destination using first the links of the first type and then those of the second type. The algorithms we develop will use such a strategy. They give an optimal centralized algorithm for the permutation routing problem in full-duplex 2-circulant graphs and double-loop networks (i.e. oriented 2-circulant graphs).

Another network of great practical importance is the *double-loop* network : a network modeled by a graph with vertex set $v = \{v_0, \ldots, v_{n-1}\}$ such that there are two integers h_1 and h_2 such that $E = \{v_i v_{i \pm h_1}, v_i v_{i \pm h_2}\}$. The permutation routing problem in this network is studied in [14]. The authors gave an algorithm for the permutation routing problem which in mean uses 1.12ℓ steps (the mean being empirically measured). In [15] the authors described an optimal centralized permutation routing algorithm in k-circulant graphs ($k \geq 2$), and in [42] an optimal distributed permutation routing in 2-circulant graphs was obtained.

The problem has been also studied for packets arriving dynamically. In [18], the author gave an optimal online schedule for the linear array. He also gave a 2-approximation for rings and show that, using shortest path routing, no better approximation algorithm exists. In [21], the authors studied Cube Connected Cycles : $CCC(n, 2^n)$ (hypercubes of dimension n where each node is replaced by a cycle of length n). They gave an algorithm working in $\mathcal{O}(n^2)$ with $\mathcal{O}(1)$ buffers for the online partial permutation routing (PPR).

1.3.2 Plane grids

Maybe the most studied networks in the literature are the two dimensional grids (or plane grids), and among them in particular the square grid has deserved special attention. Let us briefly overview what has been previously done on (ℓ, k)-routing in plane grids.

In [32] the first optimal permutation routing (with running time $2n-2$), and queues of size 1008 appears. The queue size is reduced in [41] to 112 and further in [48] to 81. Furthermore, in [48] the authors provide another algorithm running in near-optimal time $2n + \mathcal{O}(1)$ steps with a maximum queue size of only 12. [36] gives an *asymptotically* optimal algorithm for $(1, k)$-routing on plane grids, with queues of small constant size. They introduced for the first time the $(1, k)$-routing and the $(1, any)$-routing problems. This result was further improved in [47], where the authors gave near-optimal deterministic algorithm running in $\sqrt{k}\frac{n}{2} + \mathcal{O}(n)$ steps. They gave another algorithm, slightly worse in terms of number of steps, but with queues of size only 3. They also studied the general problem of (ℓ, k)-routing in square grids. They proposed lower bounds and near-optimal randomized and deterministic algorithms. They finally extended them to higher dimensional meshes. They performed (ℓ, ℓ)-routing in $\mathcal{O}(\ell n)$ steps, the lower bound being $\Omega(\sqrt{\ell k n})$ for (ℓ, k)-routing. Finally, in [39], the authors gave deterministic and randomized algorithms for (ℓ, k)-routing in square grids, with constant queue size. The running time is $\mathcal{O}(\sqrt{\ell k n})$ steps, which is optimal according to the bound of [47]. This work closed a gap in the literature, since optimal algorithms were only known for $\ell = 1$ and $\ell = k$.

Nodes in a hexagonal network are placed at the vertices of a regular triangular tessellation, so that each node has up to six neighbors. In other words, a hexagonal network is
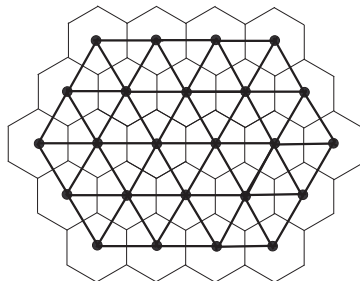
FIG. 1 – Hexagonal network ($\triangle$) and hexagonal tessellation ($\bigcirc$)

a finite subgraph of the triangular grids. These networks have been studied in a variety of contexts, specially in wireless and interconnection networks. The most known application may be to model cellular networks with hexagonal networks where nodes are base stations. But these networks have been also applied in chemistry to model benzenoid hydrocarbons [52, 25], in image processing and computer graphics [26].

In a radiocommunication wireless environment [37], the interconnection network among base stations constitutes a hexagonal network, i.e. a triangular grid, as it is shown in FIG. 1.

Tessellation of the plane with hexagons may be considered as the most natural because cells have optimal diameter to area ratio. Hexagonal networks are finite subgraphs of the triangular grid. The triangular grid can also be obtained from the basic 4-mesh by adding NE to SW edges, which is called a 6-mesh in [53]. Here we study convex subgraphs, i.e. that contain all shortest paths between all pairs of nodes, of the square, triangular and hexagonal grid. Summarizing, to the best of our knowledge the only optimal algorithms concerning (ℓ, k)-routing on plane grids (according to the lower bound of [47]) have been found on square grids, but modulo a constant factor [39]. On triangular and hexagonal grids, the best results are randomized algorithms with good performance [46].

1.4 Our Contribution

In this paper we study the permutation routing, r-central and (ℓ, k)-routing problems on plane grids, that is square grids, triangular grids and hexagonal grids. We use the *store-and-forward* Δ-port model, and we consider both full and half-duplex networks.

We have seen in Section 1.3.2 that the only plane grid for which there existed an optimal (ℓ, k)-routing is the square grid. In addition, what is important is that the results of these articles concerning (ℓ, k)-routing in plane grids are optimal modulo a constant factor. In this paper we improve these results by giving tight algorithms including the constant, in the cases of square, triangular and hexagonal grids. It is important to stress that all the algorithms presented in this paper are **distributed** (except the one given in Appendix B),

i.e. can be implemented independently at each node. Our algorithms only use shortest paths, therefore they achieve minimum stretch. In addition, the algorithms are oblivious, so they can be used in an on-line scenario, unless the performance guarantees that we prove apply only to the off-line case. The main new results of this article are the following :

1. First tight (also including the constant factor) permutation routing algorithms in full-duplex hexagonal grids, and half duplex triangular and hexagonal grids.
2. First tight (also including the constant factor) r-central routing algorithms in triangular and hexagonal grids.
3. First tight (also including the constant factor) (k, k)-routing algorithms in square, triangular and hexagonal grids.
4. Good approximation algorithms for (ℓ, k)-routing in square, triangular and hexagonal grids.

This paper is structured as follows. In Section 2 we study the permutation routing problem. Although permutation routing had already been solved for square grids, we begin in Section 2.1 by illustrating our algorithm for such grids. Then in Section 2.2 we give a tight permutation routing algorithm for half-duplex triangular grids, using the optimal algorithm of [44]. In Section 2.3 we provide a tight permutation routing algorithm for full-duplex hexagonal grids and a tight permutation routing algorithm for half-duplex hexagonal grids. In Section 3 we focus on $(1, any)$-routing, giving an optimal r-central routing algorithms for the three types of grids. We finally move in Section 4 to the general (ℓ, k)-routing problem. We provide a distributed algorithm for (ℓ, k)-routing in any grid, using the ideas of the optimal algorithm for permutation routing. We also prove lower bounds for the worst-case running time of any algorithm using shortest path routing. In addition, these lower bounds allow us to prove that our algorithm turns out to be tight when $\ell = k$, yielding in this way a tight (k, k)-routing algorithm in square, triangular and hexagonal grids. We also propose in Appendix B an approach to (ℓ, k)-routing in terms of a graph coloring problem : the WEIGHTED BIPARTITE EDGE COLORING. We give a centralized algorithm using this reduction.

2   Permutation Routing

As we have already said in Section 1, in the permutation routing problem, each processor is the origin of at most one packet and the destination of no more than one packet. The goal is to minimize the number of time steps required to route all packets to their respective destinations. It corresponds to the case $\ell = k = 1$ of the general (ℓ, k)-routing problem. This problem has been studied in a wide diversity of scenarios, such as Mobile Ad Hoc Networks [23], Cube-Connected Cycle (CCC) Networks [21], Wireless and Radio Networks [10], All-Optical Networks [33] and Reconfigurable Meshes [9].

In a grid with full-duplex links an edge can be crossed simultaneously by two messages, one in each direction. Equivalently, each edge between two nodes u and v is made of two independent arcs $\{uv\}$ and $\{vu\}$, as illustrated in Fig. 2a.
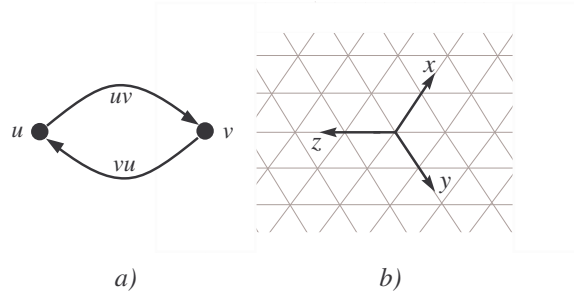
$a)$ $b)$

FIG. 2 − **a)** Each edge consists of two independent links. **b)** Axis used in a triangular grid

Remark 2.1 *If the network is half-duplex, it is easy to construct a 2-approximation algo-rithm from an optimal algorithm for the full-duplex case by introducing* odd-even *steps, as explained for example in [14].*

2.1   Square grid

Many communication networks are represented by graphs satisfying the following prop-erty : for any pair of nodes u and v, the edges of a shortest path from u to v can be partitioned into k disjoint classes according to a well-defined criterium. For instance, on a triangular grid the edges of a shortest path can be partitioned into *positive* and *negative* ones [44]. Similarly, on a k-circulant graph the edges can be partitioned into k classes according to their length.

In graphs that satisfy this property there exists a natural routing algorithm : route all packets along one class of edges after another. For hexagonal networks this algorithm turns out to be optimal [44]. Optimality for 2-circulant graphs is proved using a static approach in [19], and recently using a dynamic distributed algorithm in [42]. In [19] the authors introduce the notion of *big-foot* algorithms because their algorithm routes packets first along *long hops* and then along *short hops* in a 2-circulant graph.

On the square grid, the *big-foot* algorithm consists of two phases, moving each packet first horizontally and then vertically. In this way a packet may wait only during the second phase. Using that all destinations are distinct, the optimality for square grid is easy to prove. Summarizing, it can be proved that

Theorem 2.1 *There is a translation invariant oblivious optimal permutation routing algo-rithm for full-duplex networks that are convex subgraphs of the infinite square grid.*

2.1.1   Regarding the queue size

Of course, this is not the first optimal permutation routing result on square grids, as the classical $x - y$ routing (first route packets through the horizontal axis, and then through the vertical axis) has been used for a long time. Thus, another more challenging issue is to reduce the queue size, as we have already discussed in Section 1.3.2. Leighton describes in [30] a simple off-line algorithm for solving any permutation routing problem in $3n - 3$ steps on a $n \times n$ square grid, using queues of size one. Since the diameter of a $n \times n$ square grid is $2n-2$, this algorithm provides a $\frac{3}{2}$-approximation. The main drawback is that this algorithm is off-line and centralized. In contrast, our oblivious distributed algorithm is optimal in terms of running time, but it is easy to see that on a $n \times n$ square grid, the queue size can be $\frac{n-1}{2}$. Up to date, the best algorithm running in optimal time to route permutation routing instances on square grids is the algorithm of Sibeyn et al. [48], using queues of size 81. So far, there is no algorithm that guarantees optimal running time with queues of size 1, and it is unlikely that such an algorithm exists.

Remark 2.2 *The same observation regarding the unbounded queue size applies to all the algorithms described in this article. However, our aim is to match the optimal running time, rather than minimizing the queue size. Additionally, it turns out that some appropriate modifications of the permutation routing algorithms that we provide for plane grids allow us to find oblivious algorithms which route any permutation within a factor 3 of the optimal running time, and using queues of size 1 (in fact, we can say something stronger : we just need memory to keep 1 message at each node). We do not describe these modifications in this article.*

2.2   Triangular grid

We use the addressing scheme introduced in [37] and used also in [44] : we represent any address on a basis consisting of three unitary vectors $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ on the directions of three axis x, y, z with a 120 degree angle among them, intersecting on an arbitrary (but fixed) node O . This node is the origin and is given the address $\boldsymbol{O} = (0, 0, 0)$. This basis is represented in Fig. 2b. Thus, we can assume that each node $P \in V$ is labeled with an address $\boldsymbol{P} = (P_1, P_2, P_3)$ expressed in this basis $\{\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$ with respect to the origin O. At the beginning, each node S knows the address of the destination node D of the message placed initially at S, and computes the relative address $\overrightarrow{SD} = \boldsymbol{D} - \boldsymbol{S}$ of the message. Note that this relative address does not depend on the choice of the origin node O. This relative address is the only information that is added in the heading of the message to be transmitted, constituting in this way the packet to be sent through the network.

Using that $\boldsymbol{i} + \boldsymbol{j} + \boldsymbol{k} = 0$, it is easy to see that if (a, b, c) and (a', b', c') are the relative addresses of two packets, then $(a, b, c) = (a', b', c')$ if and only if there exists $d \in \mathbb{Z}$ such that $a' = a + d$, $b' = b + d$, and $c' = c + d$.

We say that an address $\overrightarrow{SD} = (a, b, c)$ is of the *shortest path form* if there is a path from node S to node D, consisting of a units of vector $\boldsymbol{i}$, b units of vector $\boldsymbol{j}$ and c units of vector $\boldsymbol{k}$, and this path has the shortest length.

Theorem 2.2 ([37]) *An address (a, b, c) is of the shortest path form if and only if at least one component is zero, and any two components do not have the same sign.*

Corollary 2.1 ([37]) *Any address has a unique shortest path form.*

Thus, each address $\overrightarrow{SD}$ written in the shortest path form has at most two non-zero components, and they have different sign. In fact, it is easy to find the shortest path form using the next result.

Theorem 2.3 ([37]) *If $\overrightarrow{SD} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$, then*

$$|\overrightarrow{SD}| = \min(|a - c| + |b - c|, |a - b| + |b - c|, |a - b| + |a - c|).$$

Permutation routing on full-duplex triangular grids has been solved recently [44] attaining the distance lower bound of $\ell_{\max}$ routing steps, where $\ell_{\max}$ is the maximum length over the shortest paths of all packets to be sent through the network.

As said in Remark 2.1, ff the network is half-duplex, one can construct a 2-approximation algorithm from an optimal algorithm for the full-duplex case by introducing *odd-even* steps. Thus, using this algorithm we obtain an upper bound of $2\ell_{\max}$ for half-duplex triangular grids.

Let us show with an example that this naïve algorithm is tight. That is, we shall give an instance requiring at least $2\ell_{\max}$ running steps, implying that no better algorithm for a general instance exists. Indeed, consider a set of nodes distributed along a line on the triangular grid. We fix $\ell_{\max}$ and an edge e on this line, and put $\ell_{\max}$ packets at each side of e along the line, at distance at most $\ell_{\max} - 1$ from an end-vertex of e. For each packet, each destination is placed at the other side of e with respect to its origin, at distance exactly $\ell_{\max}$ from the origin. It is easy to check that the congestion of e (that is, the number of shortest paths containing e) is $2\ell_{\max}$, and thus any algorithm using shortest path routing cannot perform in less than $2\ell_{\max}$ steps. On the other hand, $\ell_{\max}$ is a lower bound for any distance, yielding that the approximation ratio of our algorithm is at most 2.

Previous observations allow us to state the next result :

Theorem 2.4 *There exists a tight permutation routing algorithm for half-duplex triangular grids performing in at most $2\ell_{\max}$ steps, where $\ell_{\max}$ is the maximum length over the shortest paths of all packets to be sent. This algorithm is a 2-approximation algorithm for a general instance.*

2.3 Hexagonal grid

In a hexagonal grid one can define three types of *zigzag chains* [50], represented with thick lines in FIG. 3. Similarly to the triangular grid, in the hexagonal grid any shortest path between two nodes uses at most two types of zigzag chains [50]. Let us now give a lower bound for the running time of any algorithm. Consider the edge labeled as e in FIG. 3, and the two chains containing it (those shaping an X). Fix $\ell_{\max}$ and e, and put one message on all nodes placed at both chains at distance at most $\ell_{\max} - 1$ from an endvertex of e. As in

the case of the triangular grid, choose the destinations to be placed on the other side of e along the same zigzag chain than the originating node, at distance exactly $\ell_{\max}$ from it. It is clear that all the shortest paths contain e. It is also easy to check that the congestion of e is in this case $4\ell_{\max} - 4$, constituted of symmetric loads $2\ell_{\max} - 2$ in each direction of e. Thus, $2\ell_{\max} - 2$ establishes a lower bound for the running time of any algorithm in the full-duplex case, whereas $4\ell_{\max} - 4$ is a lower bound for the half-duplex case, under the assumption of shortest path routing.
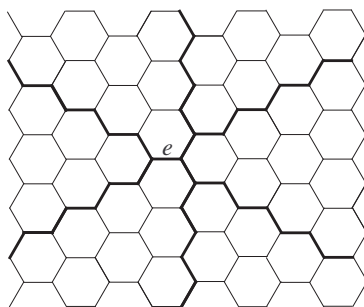

FIG. 3 – Zigzag chains in a hexagonal grid

Let us now describe a routing algorithm which reaches this bound. We have three types of edges according to the angle that they form with any fixed edge. Each edge belongs to exactly two different chains, and conversely each chain is made of two types of edges. Moreover, in an infinite hexagonal grid any 2 chains of different type intersect exactly on one edge.

Given a pair of origin and destination nodes S and D, it is possible to express the relative address $D - S$ counting the number of steps used by a shortest paths on each type of chain. In this way we obtain an address $D - S = (a, b, c)$ on a generating system made of unitary vectors following the directions of the three types of chains (it is not a basis in the strict sense, since these vectors are not linearly independent on the plane. However, we will call it so). Choose this basis so that the three vectors form angles of 120 degrees among them. As it happens on the triangular grid [37, 43], there are at most two non-zero components (see [50]), and in that case they must have different sign. Nevertheless, now the address is not unique, since an edge placed at the *bent* (that is, a change from a type of chain to another) of a shortest path is part of both types of chains. Anyway, this ambiguity is not a problem in the algorithm that propose, as we will see below.

Suppose first that edges are bidirectional or, said otherwise, full-duplex. Roughly, the idea is to use the optimal algorithm for triangular grids described in [43], and adapt it to hexagonal grids. For that purpose we label the three types of zigzag chains c_1, c_2, c_3, and the three types of edges e_1, e_2, e_3. Without loss of generality, we label them in such a way that c_1 uses edges of type e_2 and e_3, c_2 uses e_1 and e_3, and c_3 uses e_1 and e_2 (see FIG. 4).
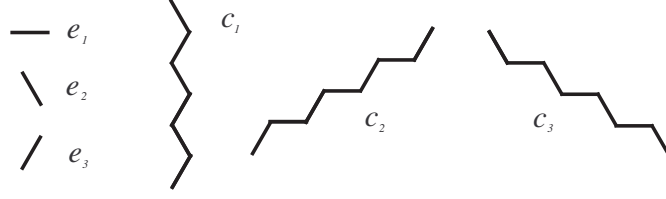
FIG. 4 − 3 types of chains and edges in a hexagonal grid

For each type of edge, we define two phases according to the type of chain that uses this type of edge. This defines two global phases, namely : during Phase 1, c_1 uses e_2, c_2 uses e_3, and c_3 uses e_1. Conversely, during Phase 2 c_1 uses e_3, c_2 uses e_1, and c_3 uses e_2.

We suppose that at each node packets are grouped into distinct queues according to the next edge (according to the rules of the algorithm) along its shortest path. Given the relative addresses $D - S$ in the form (a, b, c), the algorithm can be described as follows.

At each node u of the network :

1) During the first step, move all packets along the direction of their negative component. If a packet's address has only a positive component, move it along this direction.

2) From now on, change alternatively between Phase 1 and Phase 2. At each step (the same for both phases) :

    a) If there are packets with negative components, send them immediately along the direction of this component.

    b) If not, for each outgoing edge order the packets in decreasing number of remaining steps, and send the first packet of each queue.

3) If at some point, all the packets in u have remaining distance one, send them immediately.

Let us analyze the correctness and optimality of this algorithm.

In **1)** all packets can move, since initially there is at most one packet at each node. In **2a)**, there can only be one packet with negative component at each outgoing edge [43]. In **2b)** *the* packet with maximum remaining length at each outgoing edge is unique, since all these packets are moving along their last direction (their negative component is already finished, otherwise they would be in **2a)**) and each node is the destination of at most one packet. Hence, using this algorithm every 2 steps (one of Phase 1 and one of Phase 2) the maximum remaining distance over all packets decreases by one. Moreover, during the first step all packets decrease their remaining distance by one. Because of this, after the $(2\ell_{\max} - 3)th$ step the maximum remaining distance has decreased at least $1 + \frac{2\ell_{\max} - 4}{2} = \ell_{\max} - 1$ times, hence the maximum remaining distance is 1, and we are in **3)**. Since all destinations are different, all packets can reach simultaneously their destinations. Thus, the total running time is at most $2\ell_{\max} - 3 + 1 = 2\ell_{\max} - 2$, meeting the *worst case* lower bound. Again, $\ell_{\max}$

is a lower bound for any instance, hence the algorithm constitutes a 2-approximation for a general instance.

Theorem 2.5 *There exists a tight permutation routing algorithm for full-duplex hexagonal grids performing in $2\ell_{\max} - 2$ steps, where $\ell_{\max}$ is the maximum length over the shortest paths of all packets to be sent.*

Remark 2.3 *The optimality stated in Theorem 2.5 is true only under the assumption of shortest path routing. This means that for certain traffic instances the total deliver time may be shorter if some packets do not go through their shortest path. To illustrate this phenomenon, consider the example of* FIG. 5. *Node labeled i wants to send a message to node labeled i', for $i = 1, \ldots, 8$. We have that $\ell_{\max} = 5$, and thus our algorithm performs in $2\ell_{\max} - 2 = 8$ steps. It is clear that all shortest paths use edge e, and its congestion bottlenecks the running time. Suppose now that we route the messages originating at even nodes through the path defined by the edges $\{abcd\}$, instead of $\{fe\}$, and keep the shortest path routing for messages originating at odd nodes. One can check that with this routing only 7 steps are required.*
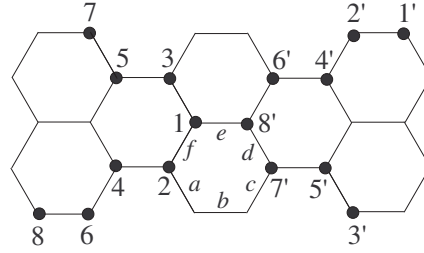

FIG. 5 – Shortest path is not always the best choice

In the half-duplex case, just introduce again odd-even steps in both phases. Thus, we have Phase 1-even, Phase 1-odd, Phase 2-even, and Phase 2-odd, which take place sequentially. Now, **1)** consists obviously of two steps (even/odd). Using this algorithm, every 4 steps the maximum remaining distance decreases by one. In addition, during the first 2 steps and during the last 2 steps all packets decrease their remaining distance by one. Thus, the total running time is at most twice the time of the full-duplex case, that is $2(2\ell_{\max} - 2) = 4\ell_{\max} - 4$ steps, meeting again the lower bound for the running time of any routing algorithm using shortest path routing. Again, this algorithm constitutes a 4-approximation for a general instance.

Theorem 2.6 *There exists an tight permutation routing algorithm for half-duplex hexagonal grids performing in $4\ell_{\max} - 4$ steps, where $\ell_{\max}$ is the maximum length over the shortest paths of all packets to be sent.*

Remark 2.4 *As explained in Appendix A, there exists an embedding of the triangular grid into the hexagonal grid with load, dilation, and congestion 2. Using this embedding, any algorithm performing on k steps on the triangular gird performs on $2k$ steps on the hexagonal grid. Using this fact, we obtain a permutation routing algorithm on full-duplex hexagonal grids performing on $2\ell_{\max}$ steps. Note that the optimal result given in Theorem 2.5 is slightly better.*

The same applies to half-duplex hexagonal networks, with a running time of $4\ell_{\max}$ using the embedding, in comparison to $4\ell_{\max} - 4$ steps given by Theorem 2.6.

3 $(1, any)$-**Routing**

In this case the routing model is the following : each packet has at most one packet to send, but there are no constraints on the destination. That is, in the worst case all packets can be sent to one node. This special case where all packets want to send a message to the same node in often called *gathering* in the literature [5]. Notice that this routing model is conceptually different from the $(1, k)$-routing, where the maximum number of packets that a node can receive is fixed *a priori*.

Square grid   Assume first that edges are bidirectional. The modifications for the half-duplex case are similar to those explained in the previous section.

We will focus on the case where all packets surrounding a given vertex want to send a packet to that vertex. We call this situation *central* routing, and if we want to specify that all nodes at distance at most r from the center want to send a packet, we note it as *r-central* routing. Note that this situation is realistic in many practical applications, since the central vertex can play the role of a router or a gateway in a local network.

Lemma 3.1 (Lower Bound) *The number of steps required in a r-central routing is at least $\binom{r+1}{2}$.*

Proof: Let us use the bisection bound [17] to prove the result. It is easy to count the number of points at distance at most r from the center, which is $4\binom{r+1}{2}$. Now consider the cut consisting of the four edges outgoing from the central vertex. All packets must traverse one of these edges to arrive to the central vertex. This cut gives the bisection bound of $4\binom{r+1}{2}/4$ routing steps.   □

Let us now describe an algorithm meeting the lower bound.

Proposition 3.1 *There exists an optimal r-central routing algorithm on square grids performing in $\binom{r+1}{2}$ routing steps.*

Proof: Express each node address in terms of the relative address with respect to the central vertex. In this way each node is given a label (a, b). Then, for each packet placed in a node with label (a, b) our routing algorithm performs the following :

- If $ab = 0$, send the packet along the direction of the non-zero component.

- If $ab > 0$, send the packet along the vertical axis.
- If $ab < 0$, send the packet along the horizontal axis.

Queues are managed so that the packets having greater remaining distance have priority.

This routing divides the square grid into 4 subregions surrounding the central vertex, as shown in FIG. 6. The type of routing performed in each subregion is symbolized by an arrow.
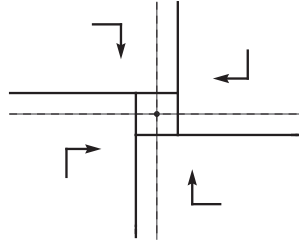

FIG. 6 − Division of the grid in the proof of Proposition 3.1

Let us now compute the running time in the r-central case. It is obvious that using this algorithm all packets are sent to the 4 axis outgoing from the central vertex. The congestion of the edge in the axis containing the central vertex along each line is $1+2+3+\ldots+r = \binom{r+1}{2}$. Since at each step one packet reaches its destination along each line, we conclude that $\binom{r+1}{2}$ is the total running time of the algorithm. $\qquad\square$

Triangular grid   The same idea of the square grid applies to the triangular grid. In this case, the number of nodes at distance at most r is $6\binom{r+1}{2}$. The cut is made of 6 edges. Dividing the plane onto 6 subregions gives again an optimal algorithm performing in $\binom{r+1}{2}$ steps.

Hexagonal grid   The same idea gives an optimal routing in the r-central case. In this case the degree of each vertex is 3, and then it is easy to check (maybe a drawing using FIG. 3 can help) that there are $3\binom{r+1}{2}$ nodes at distance at most d that may want to send a message to the central vertex, and the cut has size 3. As expected, the running time is again $\binom{r+1}{2}$.

4 (ℓ, k)-Routing

Recall that in the general (ℓ, k)-routing problem each node can send at most ℓ packets and receive at most k packets. We propose a distributed approximation algorithm using

the ideas of the algorithms that we have developed for the permutation routing problem. We also provide lowers bounds for the running time of any algorithm using shortest path routing, that allow us to prove that our algorithm is tight when $\ell = k$, on any grid.

Remark 4.1 *We also propose in Appendix B an approach to find a solution of the (ℓ, k)-routing problem, on any grid, using the problem of* WEIGHTED EDGE COLORING *in a bipartite graph. Nevertheless, the algorithm obtained using this approach is centralized.*

We start by describing the results for full-duplex triangular grids. The results can be completely adapted to square grids, but we focus on the other grids since there were few results in the literature. We also show how to adapt the results to hexagonal grids and to the half-duplex version. In this section we denote $c := \left\lceil \frac{\max\{\ell, k\}}{\min\{\ell, k\}} \right\rceil = \left\lceil \max\{\frac{\ell}{k}, \frac{k}{\ell}\} \right\rceil$. Note that $c \geq 1$. Lemma 4.1 and Lemma 4.2 provide two lower bounds for the running time of any algorithm using shortest paths.

Lemma 4.1 (First lower bound) *The worst-case running time of any algorithm for (ℓ, k)-routing on full-duplex triangular grids using shortest path routing satisfies*

$$Running\ time \geq \min\{\ell, k\} \cdot \ell_{\max}$$

Proof: Consider a set of $\ell_{\max}$ nodes placed along a line, placed consecutively at one side of a distinguished edge e. Each node wants to send $\min\{\ell, k\}$ messages to the nodes placed at the other side of e along the line, at distance $\ell_{\max}$ from it. Then the congestion of e is $\min\{\ell, k\} \cdot \ell_{\max}$, giving the bound. $\qquad\qquad\square$

Definition 4.1 *Given a vertex v, we call the rectangle of side (a, b) starting at v the set $R_{a,b}^v = \{v + \alpha\mathbf{i} + \beta\mathbf{j}, 0 \leq \alpha < a, 0 \leq \beta < b\}$ We call such a rectangle a square if $a = b$. Notice that in the triangular grid the node set is generated by $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$, where $\mathbf{k} = -\mathbf{i} - \mathbf{j}$, as we have explained in Section 2.2.*

Using standard graph terminology, given a graph $G = (V, E)$ and a subset $S \subseteq V$, the set $\Gamma(S)$ denotes the (open) neighborhood in G of the vertices in S. The following theorem can be found, for example, in [13].

Theorem 4.1 (Corollary of Hall's theorem [13]) *Let $G = (V, E)$ be a bipartite graph, with $V = X \cup Y$. If for all subsets A of X, $|\Gamma(A)| \geq c|A|$, then for each $x \in X$, there exists $S_x \subset Y$ such that $|S_x| = c$, and $\forall x, x' \in X$, $S_x \cap S_{x'} = \emptyset$ and $S_x \subset \Gamma(x)$.*

We use this theorem to prove the following lower bound.

Lemma 4.2 (Second lower bound) *The worst-case running time of any algorithm for (ℓ, k)-routing on full-duplex triangular grids using shortest path routing satisfies*

$$Running\ time \geq \left\lceil \frac{\max\{\ell, k\}}{4} \cdot \left\lfloor \frac{\ell_{\max} + 1}{\sqrt{c + 1}} \right\rfloor \right\rceil ,$$

where $c = \left\lceil \frac{\max\{\ell, k\}}{\min\{\ell, k\}} \right\rceil$.

Proof: Suppose without loss of generality that $\ell \geq k$, otherwise replace ℓ by $\max\{\ell, k\}$. Let v be a vertex, and consider the square $R_{d,d}^v$, with $d := \left\lfloor \frac{\ell_{\max}+1}{\sqrt{c+1}} \right\rfloor$. We claim that all nodes inside this square can send ℓ messages such that all destination nodes are in the destination set $D = R_{d+\ell_{\max}, d+\ell_{\max}}^v \setminus R_{d,d}^v$. Let S be the subgrid generated by positive linear combinations of the vectors $\mathbf{i}$ and $\mathbf{j}$. More precisely, $S := \{v + \alpha\mathbf{i} + \beta\mathbf{j}, \alpha \geq 0, \beta \geq 0\}$. FIG. 2b gives a graphical illustration.

To prove this, we consider a bipartite graph H on vertex set $R_{d,d}^v \cup D$, with an edge between a vertex of $R_{d,d}^v$ and a vertex of D if they are at distance at most $\ell_{\max}$ in S. To apply Theorem 4.1, we have to show that any subset of vertices $A \subset R_{d,d}^v$ has at least $c|A|$ neighbors in H. Theorem 4.1 will then ensure the existence of a feasible repartition of the messages from vertices of $R_{d,d}^v$ to those of D such that they all travel a distance at most $\ell_{\max}$.

Given $A \subset R_{d,d}^v$, let us call $D_A := \{u \in D : \mathrm{dist}_S(A, u) \leq \ell_{\max}\}$, where $\mathrm{dist}_S(A, u)$ means the minimum distance in S from any vertex of A to the vertex u. For any $A \subset R_{d,d}^v$, we need to show that

$$|D_A| \geq c|A| \tag{1}$$

Without loss of generality we suppose that A is maximal, in the sense that there is no set A' strictly containing A with $D_A = D_{A'}$. Instead of considering all possible sets A, we will show below that we can restrict ourselves to rectangles. Hence given a set A, we denote by R_A the smallest rectangle containing the subset of vertices A. We first claim that

$$|D_{R_A} \setminus D_A| \leq |R_A \setminus A| \tag{2}$$

Indeed, this equality can be shown by induction on $|R_A \setminus A|$. For $|R_A \setminus A| = 0$ the equality is trivial. Suppose that it is true for $|R_A \setminus A|$. The induction step consists in showing that there is an element x in $R_A \setminus A$ such that $|D_{R_A} \setminus D_{A \cup \{x\}}| - |D_{R_A} \setminus D_A| \leq 1$ (note that $D_{R_{A \cup \{x\}}} = D_{R_A}$) :

- If there exists x such that $x + \mathbf{j}$ and $x - \mathbf{i}$ are in A and $x - \mathbf{j}$ is not in A, then we select this x. From x the only new vertex we may add to D_A is $x + \ell_{\max}\mathbf{i}$.
- Otherwise, if there exists x such that $x - \mathbf{j}$ and $x - \mathbf{i}$ are in A and $x + \mathbf{i}$ is not in A, then we select this x. In this case the only new vertex we may add to D_A is $x - \ell_{\max}\mathbf{k}$.
- If none of the previous cases holds, since R_A is the smallest rectangle containing A, and A is maximal, then necessarily there exists an x such that $x + \mathbf{i}$ and $x - \mathbf{j}$ are in A and $x - \mathbf{i}$ is not in A. We select this x, and the only new vertex we may add to D_A is $x + \ell_{\max}\mathbf{j}$.

Thus, in all cases there exists an x adding at most one neighbor to $D_{R_A} \setminus D_A$, which finishes the induction step and proves Equation (2). To finish the proof of the fact that we can

restrict ourselves to rectangles, we show that, for any subset A, if Inequality (1) holds for R_A, then it also holds for A. Indeed, Inequality (1) applied to R_A gives :

$$c|R_A| \leq |D_{R_A}| \; , \; \text{ which is equivalent to}$$

$$c(|A| + |R_A \setminus A|) \leq |D_A| + |D_{R_A} \setminus D_A| \tag{3}$$

Using Inequality (2) and the fact that $c \geq 1$, Inequality (3) clearly implies that Inequality (1) holds.

Henceforth we assume that A is a rectangle. The last simplification consists in proving that we can restrict ourselves to rectangles containing v. In other words, it will be sufficient to prove Inequality (1) for all rectangles $R_{a,b}^v$. Given a rectangle R not positioned at v, the rectangle R' of the same size positioned at v has less neighbors, hence if Inequality (1) holds for R', it also holds for R.

Finally let us prove that Inequality (1) holds for all rectangles $R_{a,b}^v$, with $1 \leq a, b < d$. We have that $|R_{a,b}^v| = ab$ and $|D_{R_{a,b}^v}| = (a + \ell_{\max})(b + \ell_{\max}) - d^2$.

By the choice of d, starting from the Inequality $d^2 \leq \frac{(\ell_{\max}+1)^2}{c+1}$ and using that $1 \leq a, b$, one obtains that $d^2 c \leq (\ell_{\max} + a)(\ell_{\max} + b) - d^2$ for any $1 \leq a, b < d$. This implies, using $a, b < d$, that $cab \leq (a + \ell_{\max})(b + \ell_{\max}) - d^2$ for any $1 \leq a, b < d$, hence Inequality (1) (i.e. $c|R_{a,b}^v| \leq |D_{R_{a,b}^v}|$) holds.

So by Theorem 4.1, each one of the d^2 nodes in $R_{d,d}^v$ can send ℓ messages to the nodes of D. Since the number of edges going from $R_{d,d}^v$ to D is $4d - 1$, we apply the bisection bound discussed in Section 1.1.1 to conclude that there is an edge of the border of the square $R_{d,d}^v$ with congestion at least $\left\lceil \frac{\ell \cdot d^2}{4d-1} \right\rceil > \left\lceil \frac{\ell \cdot d}{4} \right\rceil$. This finishes the proof of the lemma. $\qquad \square$

We observe that this second lower bound is strictly better than the first one if and only if

$$\frac{c}{\sqrt{c+1}} > \frac{4\ell_{\max}}{\ell_{\max} + 1}$$

If both c and $\ell_{\max}$ are big, the condition becomes approximately :

$$\frac{\max\{\ell, k\}}{\min\{\ell, k\}} > 16$$

That is, the second lower bound is better when the difference between ℓ and k is big. This is the case of broadcast or gathering, where messages are originated (or destined) from (or to) a small set of nodes of the network.

The two lower bounds can be combined to give :

Lemma 4.3 (Combined lower bound) *The worst-case running time of any algorithm for (ℓ, k)-routing on full-duplex triangular grids using shortest path routing satisfies*

$$Running\ time \geq \max \left(\ell_{\max} \cdot \min\{\ell, k\},\ \max\{\ell, k\} \cdot \left\lfloor \frac{\ell_{\max} + 1}{4\sqrt{c+1}} \right\rfloor \right) \approx \ell_{\max} \cdot \max \left(\min\{\ell, k\},\ \frac{\max\{\ell, k\}}{4\sqrt{c+1}} \right)$$

Now we provide an algorithm from which we derive an upper bound.

Proposition 4.1 (Upper bound (algorithm)) *The algorithm for (ℓ, k)-routing on full-duplex triangular grids is the following : route all packets as in the permutation routing case. That is, at each node send packets first in their negative component, breaking ties arbitrarily (there can be ℓ packets in conflict in a negative component). If there are no packets with negative components, send any of the (at most k) packets with maximum remaining distance.*

$$Running\ time \leq \begin{cases} \min\{\ell, k\} \cdot \frac{c(c-1)}{2} + \max\{\ell, k\} \cdot (\ell_{\max} - c + 1) & ,\ if\ c \leq \ell_{\max} \\ \min\{\ell, k\} \cdot \frac{\ell_{\max}(\ell_{\max} + 1)}{2} & ,\ if\ c > \ell_{\max} \end{cases}$$

Proof: Suppose again without loss of generality that $\ell \geq k$. We proceed by decreasing induction on $\ell_{\max}$. We prove that after $\min\{\ell, \ell_{\max}k\}$ steps, each paquet will be at distance at most $\ell_{\max} - 1$ of its destination. This yields

$$\text{Running time}(\ell_{\max}) \leq \min\{\ell, \ell_{\max}k\} + \text{Running time}(\ell_{\max} - 1)$$

$$\leq \min\{\ell, \ell_{\max}k\} + \begin{cases} \min\{\ell, k\} \cdot \frac{c(c-1)}{2} + \max\{\ell, k\} \cdot (\ell_{\max} - c) & ,\ if\ c \leq \ell_{\max} - 1 \\ \min\{\ell, k\} \cdot \frac{\ell_{\max}(\ell_{\max} - 1)}{2} & ,\ if\ c > \ell_{\max} - 1 \end{cases}$$

$$\leq \begin{cases} \min\{\ell, k\} \cdot \frac{c(c-1)}{2} + \max\{\ell, k\} \cdot (\ell_{\max} - c + 1) & ,\ if\ c \leq \ell_{\max} \\ \min\{\ell, k\} \cdot \frac{\ell_{\max}(\ell_{\max} + 1)}{2} & ,\ if\ c > \ell_{\max} \end{cases}$$

Let us consider the messages at distance $\ell_{\max}$ to their destinations. They are of two types, the one moving according to their negative component and the one moving according to their positive component.

If $c \leq \ell_{\max}$ the first ones move after at most ℓ time steps. If $c < \ell_{\max}$ they move more quickly, indeed they move at least once every $\ell_{\max}k$ steps ($\ell_{\max}k \leq c \cdot k = \ell$). This is due to the fact that when $c < \ell_{\max}$ at a given vertex, at most $\ell_{\max}k$ messages may have to move according to their negative component toward a node at distance $\ell_{\max}$.

About the messages which move according to their positive component, since a node is the destination of at most k messages, they may wait at most k steps.

Consequently, $\ell_{\max}$ decreases by at least one every $\min\{\ell, \ell_{\max}k\}$ steps, which gives the result. $\qquad\square$

This gives an algorithm which is fully distributed. Dividing the running time of this algorithm by the combined lower bound we obtain the following ratio :

$$\left\{ \begin{array}{ll} \dfrac{\min\{\ell,k\}\cdot\binom{c}{2}+\max\{\ell,k\}\cdot(\ell_{\max}-c+1)}{\ell_{\max}\cdot\max\left(\min\{\ell,k\}\;,\;\frac{\max\{\ell,k\}}{4\sqrt{c+1}}\right)} & \text{, if } c \le \ell_{\max} \\[4mm] \dfrac{\min\{\ell,k\}\cdot(\ell_{\max}+1)}{2\cdot\max\left(\min\{\ell,k\}\;,\;\frac{\max\{\ell,k\}}{4\sqrt{c+1}}\right)} & \text{, if } c > \ell_{\max} \end{array} \right.$$

We observe that in all cases the running time of the algorithm is at most $\max\{\ell,k\}\cdot\ell_{\max}$. In particular, when $\ell = k$ (that is, $c = 1$) the running time is exactly $\max\{\ell,k\}\cdot\ell_{\max} = \min\{\ell,k\}\cdot\ell_{\max}$, and therefore it is tight (see lower bound of Lemma 4.1).

Corollary 4.1 *There exists a tight algorithm for (k,k)-routing in full-duplex triangular grids.*

The previous algorithms can be generalized for half-duplex triangular grids as well as for full and half-duplex hexagonal grids. The generalization to half-duplex grids is obtained by just adding a factor 2 in both the lower bound and the running time of the algorithm, as we did for the permutation routing algorithm. Thus, let us just focus on the case of full-duplex hexagonal grids, for which we have the following theorems :

Theorem 4.2 *There exists an algorithm for (ℓ,k)-routing in full-duplex hexagonal grids whose running time is at most :*

$$\text{Running time} \le \left\{ \begin{array}{ll} 2\min\{\ell,k\}\cdot\frac{c(c-1)}{2} + 2\max\{\ell,k\}\cdot(\ell_{\max}-c+1) & \text{, if } c \le \ell_{\max} \\[2mm] 2\min\{\ell,k\}\cdot\frac{\ell_{\max}(\ell_{\max}+1)}{2} & \text{, if } c > \ell_{\max} \end{array} \right.$$

Lemma 4.4 (First lower bound) *No algorithm based on shortest path routing can route all messages using less than $2\min\{\ell,k\}\cdot\ell_{\max} - \min\{\ell,k\}$ steps in the worst case.*

Definition 4.2 *Given a vertex v, we call the rectangle of the hexagonal grid of side (a,b) starting at v to the subset of the hexagonal grid $R^v_{hexa,b} = \{v + \alpha\mathbf{i} + \beta\mathbf{j} + \gamma\mathbf{k}, 0 \le \alpha < a, -\gamma < \beta < b, 0 \le \gamma < b\} \cap H$ where H is the vertex set of the hexagonal grid. We call such a rectangle a square if $a = b$.*

The following lemma gives a second lower bound on the running time of any algorithm using shortest path routing on full-duplex hexagonal grids.

Lemma 4.5 (Second lower bound) *The worst-case running time of any algorithm using shortest path routing on full-duplex hexagonal grids satisfies :*

$$\text{Running time} \ge \left\lceil \max\{\ell,k\}(2d + \frac{d-2}{2d+1}) \right\rceil \quad,$$

where $d = \left\lfloor \frac{\sqrt{73c+64\ell_{\max}^2+121+144\ell_{\max}}}{8\sqrt{c+1}} - \frac{3}{8} \right\rfloor$ *and* $c = \left\lceil \frac{\max\{\ell,k\}}{\min\{\ell,k\}} \right\rceil$.

Notice that when $\frac{\ell_{\max}}{c}$ is big, this value tends to $2\max\{\ell,k\}\frac{\ell_{\max}}{\sqrt{c+1}}$, obtaining a performance around twice better than in triangular grids.

Proof: The proof consists in showing that the vertices of $R^v_{hexd,d}$ can simultaneously send $\max(\ell, k)$ messages to some vertices of $R^v_{hexd+\ell_{\max},d+\ell_{\max}} \setminus R^v_{hexd,d}$. This is done as for the triangular grid, using again Theorem 4.1. We do not give all the details, since the idea behind is the same as the proof of Lemma 4.2.

Since the number of vertices inside $R^v_{hexd,d}$ is $4d^2 + d - 2$, and the number of edges outgoing from $R^v_{hexd,d}$ is $2d + 1$, the congestion on these edges is $\max\{\ell, k\}\frac{4d^2+d-2}{2d+1} = \max\{\ell, k\}(2d + \frac{d-2}{2d+1})$. $\qquad\square$

5   Conclusions and Further Research

In this article we have studied the permutation routing, the r-central routing and the general (ℓ, k)-routing problems on plane grids, that is square grids, triangular grids and hexagonal grids. We have assumed the *store-and-forward* Δ-port model, and considered both full and half-duplex networks. The main new results of this article are the following :

1. Tight (also including the constant factor) permutation routing algorithms on full-duplex hexagonal grids, and half duplex triangular and hexagonal grids.
2. Tight (also including the constant factor) r-central routing algorithms on triangular and hexagonal grids.
3. Tight (also including the constant factor) (k, k)-routing algorithms on square, triangular and hexagonal grids.
4. Good approximation algorithms for (ℓ, k)-routing in square, triangular and hexagonal grids, together with new lower bounds on the running time of any algorithm using shortest path routing.

All these algorithms are completely distributed, i.e. can be implemented independently at each node. Finally, we have also formulated the (ℓ, k)-routing problem as a WEIGHTED EDGE COLORING problem on bipartite graphs.

There still remain several interesting open problems concerning (ℓ, k)-routing on plane grids. Of course, the most challenging problem seems to find a tight (ℓ, k)-routing algorithm for any plane grid, for $\ell \neq k$. Another interesting avenue for further research is to take into account the queue size. That is, to devise (ℓ, k)-routing algorithms with bounded queue size, or that optimize both the running time and the queue size, under a certain trade-off.

Références

[1] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM (JACM)*, 44(3) :486–504, 1997.

[2] J. Aspnes, C. Busch, S. Dolev, P. Fatourou, C. Georgiou, and A. Shvartsman. Eight Open Problems in Distributed Computing. *Bulletin of the EATCS no*, 90 :109–126, 2006.

[3] B. Awerbuch and Y. Azar. Local optimization of global objectives : competitive distributeddeadlock resolution and resource allocation. *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 240–249, 1994.

[4] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke. Optimal oblivious routing in polynomial time. *Journal of Computer and System Sciences*, 69(3) :383–394, 2004.

[5] J.-C. Bermond, J. Galtier, R. Klasing, N. Morales, and S. Pérennes. Hardness and approximation of Gathering in static radio networks. *Parallel Processing Letters*, 16(2) :165–183, 2006.

[6] A. Borodin and J. Hopcroft. Routing, merging and sorting on parallel models of computation. *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 338–344, 1982.

[7] C. Busch, M. Magdon-Ismail, and J. Xi. Oblivious routing on geometric networks. *Proceedings of the 17th annual ACM symposium on Parallelism in algorithms and architectures*, pages 316–324, 2005.

[8] C. Busch, M. Magdon-Ismail, and J. Xi. Optimal oblivious path selection on the mesh. *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2005), Denver, Colorado, USA, April*, 2005.

[9] J. Cogolludo and S. Rajasekaran. Permutation Routing on Reconfigurable Meshes. *Algorithmica*, 31 :44–57, 2001.

[10] A. Datta. A Fault-Tolerant Protocol for Energy-Efficient Permutation Routing in Wireless Networks. *IEEE Transactions on Computers*, 54(11) :1409–1421, November 2005.

[11] D. de Werra, M. Demange, B. Escoffier, J. Monnot, and V. T. Paschos. Weighted Coloring on Planar, Bipartite and Split Graphs : Complexity and Improved Approximation. *Lecture Notes in Computer Science*, 3341 :896–907, 2004.

[12] M. Demange, D. de Werra, J. Monnot, and V. T. Paschos. Weighted node coloring : when stable sets are expensive. *WG'02 LNCS*, 2573 :114–125, 2002.

[13] R. Diestel. *Graph Theory*. Springer-Verlag, 2005.

[14] T. Dobravec, B. Robič, and J. Žerovnik. Permutation routing in double-loop networks : design and empirical evaluation. *Journal of Systems Architecture*, 48 :387–402, 2003.

[15] T. Dobravec, J. Žerovnik, and B. Robič. An optimal message routing algorithm for circulant networks. *Journal of Systems Architecture*, 52 :298–306, 2006.

[16] P. Fraigniaud and E. Lazard. Methods and problems of communication in usual networks. *Discrete Applied Mathematics*, 53 :79–134, 1994.

[17] M. D. Grammatikakis, M. K. D. F. Hsu, and J. Sibeyn. Packet Routing in Fixed-Connection Networks : a Survey. *Journal of Parallel and Distributed Processing*, 54(2) :77–132, 1998.

[18] J. T. Havill. Online Packet Routing on Linear Arrays and Rings. *Lecture Notes in Computer Science*, 2076 :773–784, 2001.

[19] F. Hwang, T. Lin, and R. Jan. A Permutation Routing Algorithm for Double Loop Network. *Parallel Processing Letters*, 7(3) :259–265, 1997.

[20] F. Hwang, Y. Yao, and B. Dasgupta. Some permutation routing algorithms for low-dimensional hypercubes. *Theoretical Computer Science*, 270 :111–124, 2002.

[21] G. E. Jan and M.-B. Lin. Concentration, load balancing, partial permutation routing, and superconcentration on cube-connected cycles parallel computers. *Journal of Parallel and Distributed Compututing*, 65 :1471–1482, 2005.

[22] C. Kaklamanis, D. Krizanc, and T. Tsantilas. Tight bounds for oblivious routing in the hypercube. *Theory of Computing Systems*, 24(1) :223–232, 1991.

[23] D. Karimou and J. F. Myoupo. An Application of an Initialization Protocol to Permutation Routing in a Single-Hop Mobile Ad Hoc Networks. *The Journal of Supercomputing*, 31 :215–226, 2005.

[24] A. Kesselman and K. Kogan. Non-Preemptive Scheduling of Optical Switches. In *IEEE GLOBECOM*, pages 1840–1844, 2004.

[25] S. Klavžar, A. Vesel, and P. Žigert. On resonance graphs of catacondensed hexagonal graphs : structure, coding, and hamiltonian path algorithm. *MATCH Communications in Mathematical and in Computer Chemistry*, 49(49) :99–116, 2003.

[26] E. Kranakis, H. Sing, and J. Urrutia. Compas Routing in Geometric Graphs. In *11th Canadian Conference of Computational Geometry*, pages 51–54, 1999.

[27] M. Kunde, R. Niedermeier, and P. Rossmanith. Faster sorting and routing on grids with

diagonals. In *11th Symposium of Theoretical Computer Science*, 775, pages 225–236. Lecture Notes on Computer Science, 1994.

[28] F. Leighton, B. M. Maggs, and A. W. Richa. Fast Algorithms for Finding O(Congestion + Dilation) Packet Routing Schedules. In *28th Annual Hawaii International Conference on System Sciences*, pages 555–563, 1995.

[29] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet Routing and Job-Shop Scheduling in O(congestion + dilation) Steps. *Combinatorica*, 14(2) :167–186, 1994.

[30] T. Leighton. *Introduction to Parallel Algorithms and Architectures : Arrays-Trees-Hypercubes*. Morgan-Kaufman, San Mateo, California, 1992.

[31] T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 256–269, 1988.

[32] T. Leighton, F. Makedon, and I. G. Tollis. A $2n - 2$ Step Algorithm for Routing in an $n \times n$ Array with Constant-Size Queues. *Algorithmica*, 14 :291–304, 1995.

[33] W. Liang and X. Shen. Permutation Routing in All-Optical Product Networks. *IEEE Transactions on Circuits and Systems*, 49(4) :533–538, 2002.

[34] B. Maggs. A Survey of Congestion+ Dilation Results for Packet Scheduling. *40th Annual Conference on Information Sciences and Systems*, 22(24) :1505–1510, 2006.

[35] B. Maggs, F. auf der Heide, B. Vocking, and M. Westermann. Exploiting locality for data management in systems of limited bandwidth. *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 284–293, 1997.

[36] F. Makedon and A. Symvonis. Optimal algorithms for the many-to-one routing problem on 2-dimensional meshes. *Microprocessors and Microsystems*, 17 :361–367, 1993.

[37] F. G. Nocetti, I. Stojmenović, and J. Zhang. Addressing and Routing in Hexagonal Networks with Applications for Tracking Mobile Users and Connection Rerouting in Cellular Networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(9) :963–971, 2002.

[38] R. Ostrovsky and Y. Rabani. Universal O(congestion + dilation + $\log^{1+\varepsilon} N$) Local Control Packet Switching Algorithms. In *29th Annual ACM Symposium on the Theory of Computing*, pages 644–653, New York, 1997.

[39] A. Pietracaprina and G. Pucci. Optimal Many-to-One Routing on the Mesh with Constant Queues. *Lecture Notes in Computer Science*, 2150 :645–649, 2001.

[40] H. Racke. Minimizing congestion in general networks. *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 43–52, 2002.

[41] S. Rajasekaran and R. Overholt. Constant queue routing on a mesh. *Journal of Parallel and Distributed Computing*, 15 :160–166, 1992.

[42] B. Robič and J. Žerovnik. Minimum 2-terminal routing in 2-jump circulant graphs. *Computers and Artificial Intelligence*, 19(1) :37–46, 2000.

[43] I. Sau and J. Žerovnik. An Optimal Permutation Routing Algorithm for Full-Duplex Hexagonal Mesh Networks. *Preprint series, University of Ljubljana, Institute of Mathematics, Physics and Mechanics*, 44(1017), 2006. ISSN 1318–4865.

[44] I. Sau and J. Žerovnik. Optimal permutation routing on mesh networks. In *Proc. of International Network Optimization Conference*, Belgium, April 2007. 6 pages.

[45] C. Scheideler. *Universal Routing Strategies for Interconnection Networks*. Springer, 1998.

[46] J. Sibeyn. Routing on Triangles, Tori and Honeycombs. *International Journal of Fundations of Computer Science*, 8(3) :269–287, 1997.

[47] J. Sibeyn and M. Kaufman. Deterministic 1-k routing on meshes (with applications to worm-hole routing). In LNCS, editor, *11th Symposium on Theoretical Aspects of Computer Science*, volume 775, pages 237–248, 1994.

[48] J. F. Sibeyn, B. S. Chlebus, and M. Kaufmann. Deterministic Permutation Routing on Meshes. *Journal of Algorithms*, 22(1) :111–141, 1997.

[49] A. Srinivasan and C.-P. Teo. A constant-factor approximation algorithm for packet routing, and balancing local vs. global criteria. In *STOC '97 : Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 636–643, New York, NY, USA, 1997. ACM.

[50] I. Stojmenović. Honeycomb Networks : Topological Properties and Communication Algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 8(10) :1036–1042, 1997.

[51] T. Suel. Routing and Sorting on Meshes with Row and Column Buses. In *Parallel Processing Symposium*, volume Eighth International Volume, pages 411–417, 1994.

[52] R. Tošić, D. Masulović, I. Stojmenović, J. Brunvoll, B. Cyvin, and S. Cyvin. Enumeration of Polyhex Hydrocarbons up to h=17. *Journal of Chemical Information and Computer Sciences*, 35 :181–187, 1995.

[53] R. Trobec. Two-dimensional regular *d*-meshes. *Parallel Computing*, 26 :1945–1953, 2000.

[54] L. Valiant and G. Brebner. Universal schemes for parallel communication. *Proceedings*

of the thirteenth annual ACM symposium on Theory of computing, pages 263–277, 1981.

[55] R. Williams. *The Geometrical Foundation of Natural Structure : A Source Book of Design*. Dover Publications, 1979.

A　Defining the embeddings

The results already known for the square grid can be used for a triangular (resp. a hexagonal) grid if we have an adapted function mapping the square grid into the triangular (resp. hexagonal) grid. Here we propose both functions, namely *square2triangle* and *square2hexagon*.

The function *square2triangle* is illustrated in Fig. 7. We perform the same routing as in the grid, i.e. we just ignore the extra diagonal.
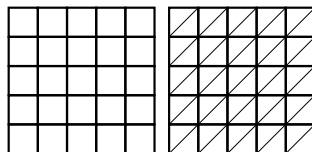
Fig. 7 – Square grid mapped into the triangular grid

In a square grid the distance between two vertices is at most twice the distance in a triangular grid. Similarly the congestion is at most doubled going from the triangular grid to the square grid. Nevertheless the maximal distance is unchanged. Indeed, the NW and SE nodes of Fig. 7 are at the same distance in both grids. Consequently, an algorithm which routes a permutation in $2n - 2$ steps is still optimal in the worst case. This is the reason why, instead of considering a square grid with one extra diagonal, we look at a triangle grid in the shape of a triangle, c.f. Fig. 8. Using the routing of the square grid in this triangle grid yields a routing within twice the optimal (i.e. minimum time in the worst case).
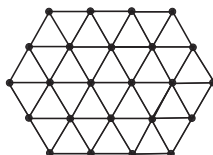

Fig. 8 – Triangular grid

The function *square2hexagon* is a little more complicated. Squares are mapped in two different ways on the hexagonal grid, as shown in Fig. 9. Some are mapped on the left side of a hexagon and some on the right side. Call them respectively *white* and *black* squares. White and black squares alternate on the grid like white and black on a chess board. The missing edge of a white square is mapped to the path of length 3 that goes on the right of the hexagon. The missing edge of a black square is mapped on the path of length 3 that goes on the right of the hexagon. In this way each edge of the square grid is uniquely mapped and each edge of the hexagonal grid is the image of exactly two edges of the square grid.
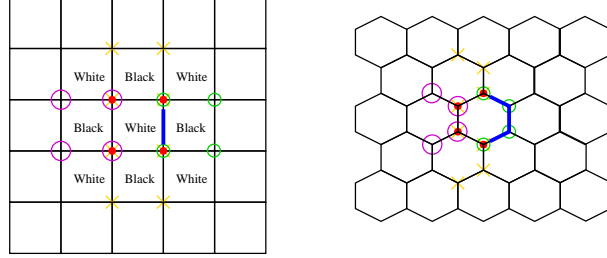
FIG. 9 – Square grid mapped into the hexagonal grid

The distance between 2 vertices in the hexagonal grid is twice the distance in the square grid plus one. Also when we adapt a routing from the square grid to the hexagonal grid using the function *square2hexagon*, the congestion may double since each edge of the hexagonal grid is the image of two edges of the square grid. Consequently, a routing obtained using the function *square2hexagon* will be within a constant multiplicative factor of the optimal.

B   An Approach for (ℓ, k)-routing Using Weighted Coloring

In any physical topology, we can represent a given instance of the problem in the following way. Given a network on n nodes, we build a bipartite graph H with a copy of each node at both sides of the bipartition. We add an edge between u and v whenever u wants to send a message to v, and assign to each edge uv a weight $w(uv)$ equal to the length of a shortest path from u to v on the original grid. In this way we obtain an edge-weighted bipartite graph H on $2n$ nodes. Note that the maximum degree of H satisfies $\Delta \leq \max\{\ell, k\}$. An example for $\ell = 2$ and $k = 3$ is depicted in FIG. 10.

The key idea behind this construction is that each matching in H corresponds to an instance of a permutation routing problem. Hence, it can be solved optimally, as we have proved for all types of grids in Section 2. For each matching M_i, we define its cost as $c(M_i) := \max\{w(e)|e \in M_i)\}$. We assign this cost because on all grids the running time of the permutation routing algorithms we have described are proportional to the length of the longest shortest path (with equality on full-duplex triangular grids).

From the classical Hall's theorem we know that the edges of a bipartite graph can be partitioned into Δ disjoint matchings (that is, a coloring of the edges), Δ being the maximum degree of the graph. In our case we have $\Delta = \max\{\ell, k\}$. Thus, the problem consists in partitioning the edges of H into Δ matchings $M_1, \ldots, M_\Delta$, in such a way that $\sum_{i=1}^{\Delta} c(M_i)$ is minimized. That is, our problem, namely WEIGHTED BIPARTITE EDGE COLORING, can be stated in the following way :
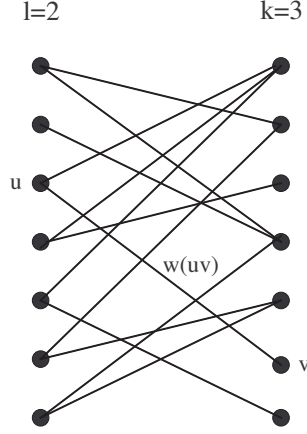
FIG. 10 − Bipartite graph modeling a $(2,3)$-routing instance

WEIGHTED BIPARTITE EDGE COLORING
Input : An edge-weighted bipartite graph H.
Output : A partition of the edges of H into matchings $M_1, \ldots, M_\Delta$, with $c(M_i) := \max\{w(e)|e \in M_i)\}$.
Objective : $\min \sum_{i=1}^{\Delta} c(M_i)$.

Therefore, $\min \sum_{i=1}^{\Delta} c(M_i)$ is the running time for routing an (ℓ, k)-routing instance using this algorithm.

Unfortunately, in [11] WEIGHTED EDGE COLORING is proved to be strongly NP-complete for bipartite graphs, which is the case we are interested in. In fact, the problem remains strongly NP-complete even restricted to cubic and planar bipartite graphs. Concerning approximation results, the authors [11] provide an inapproximability bound of $\frac{7}{6} - \varepsilon$, for any $\varepsilon > 0$. Furthermore, they match this bound with an approximation algorithm within $7/6$ on graphs with maximum degree 3, improving the best known approximation ratio of $5/3$ [12]. In [24] this innaproximability bound is proved independently on general bipartite graphs. Thus, if $\max\{\ell, k\} \leq 3$ we can find a solution of WEIGHTED BIPARTITE EDGE COLORING within $\frac{7}{6}$ times the optimal solution, and this will be also a solution for the (ℓ, k)-routing problem.

Remark B.1 *Although of theoretical value, the main problem of this algorithm is that finding these matchings is a **centralized** task. In addition, the* true *ratio, i.e. related to the optimum of the (ℓ, k)-routing, should be proved to provide an upper bound for the running time of this algorithm.*