

A self-stabilizing algorithm for the median problem in partial rectangular grids and their relatives¹

VICTOR CHEPOI, TRISTAN FEVAT, EMMANUEL GODARD AND YANN VAXÈS

LIF-Laboratoire d'Informatique Fondamentale de Marseille,
CNRS (UMR 6166), Université d'Aix-Marseille, Marseille, France
`{chepoi,fevat,godard,vaxes}@lif.univ-mrs.fr`

Abstract. Given a graph $G = (V, E)$, a vertex v of G is a *median vertex* if it minimizes the sum of the distances to all other vertices of G . The median problem consists of finding the set of all median vertices of G . In this note, we present self-stabilizing algorithms for the median problem in partial rectangular grids and relatives. Our algorithms are based on the fact that partial rectangular grids can be isometrically embedded into the Cartesian product of two trees, to which we apply the algorithm proposed by Antonoiu, Srimani (1999) and Bruell, Ghosh, Karaata, Pemmaraju (1999) for computing the medians in trees. Then we extend our approach from partial rectangular grids to a more general class of plane quadrangulations. We also show that the characterization of medians of trees given by Gerstel and Zaks (1994) extends to cube-free median graphs, a class of graphs which includes these quadrangulations.

1 Introduction

Given a connected graph G one is sometimes interested in finding the vertices minimizing the total distance $\sum_u d(u, x)$ to the vertices u of G , where $d(u, x)$ is the distance between u and x . A vertex x minimizing this expression is called a *median (vertex)* of G . The median problem consists of finding the set of all median vertices. The median problem arises with one of the basic models in discrete facility location [59] and with majority consensus in classification and data analysis [10, 18]. This is also a classical topic in graph theory [19, 59]. Linear time algorithms for computing medians are known for several classes of graphs: among them are trees, planar quadrangulations and triangulations with degree-constraints, partial rectangular grids [28], and a few other classes of graphs. Medians of polyominoes were used in [15, 30] to reconstruct special discrete sets from their vertical and horizontal projections. A distributed algorithm for computing medians in graphs is given in [52]. Finally notice that some papers

¹An extended abstract of this paper appeared in the proceedings of the 14th International Colloquium on Structural Information and Communication Complexity, SIROCCO'07. The first and the fourth authors were partly supported by the ANR grant BLAN06-1-138894 (projet OPTICOMB). The second and the third authors were supported by the ACI grant "Jeunes Chercheurs"(projet TAGADA).

[10, 53, 58] investigate the structural properties of median sets in special classes of graphs, in particular in median graphs.

In distributed systems, the median is a suitable location for information exchange and communication. Indeed, to place a common resource at a median site minimizes the cost of sharing the resource with other sites. Note also that [41] shows that, given a tree-network, choosing a median and then routing all the information through it minimizes the number of messages sent during any execution of any distributed sorting algorithm. Moreover, partial rectangular grids and trees are among the most used topologies in the design of microprocessors and distributed architectures. It is therefore of important practical interest to solve the median problem in a distributed setting on such a topology.

A distributed system can be defined as a set of processors exchanging information between neighbors. The system state, called *global state* or *configuration*, is the union of all the local states. A processor has only a local knowledge of the system, this knowledge varying according to the system connectivity. It is often desirable to maintain the system in a certain set of states, *the legitimate states*. An algorithm running in a system is said to be *self-stabilizing* if any execution has a suffix in the set of the legitimate states [35]. Self-stabilization is very desirable and useful in distributed systems because it provides immunity to transient failures and can even make possible in some cases a dynamical and transparent modification of the system topology. Besides, self-stabilizing algorithms are often elegant and simple. The self-stabilizing paradigm was introduced by Dijkstra in 1973 [34]. He gave three self-stabilizing mutual exclusion algorithms on rings, opening a field of research still extremely dynamic today in distributed calculus. Self-stabilizing algorithms have been conceived to answer problems of routing [32, 33], synchronization [7, 8, 49], leader election [4, 36], spanning tree construction [2, 5], maximum flow and maximum matching [42, 47], graph coloring [43], and mutual exclusion in several types of networks [55, 6].

Antonoiu, Srimani [3] and Bruell et al. [21] proposed a strikingly simple and nice self-stabilizing algorithm for computing the median set of a tree T . The state of each vertex is an integer s . At each step, the algorithm updates the s -value of the currently active vertex v : it sets $s(v) = 1$ if v is a leaf, otherwise it computes the sum of the s -values of all neighbors of v minus the largest s -value of a neighbor and then adds 1 (denoted by $1 + \sum(N_s^-(v))$). If the current s -value of v is different from $1 + \sum(N_s^-(v))$, then this value becomes the new s -value of v . The algorithm terminates when there are no more s -values to modify. Interestingly, this turns out to be a “valid” global state because the medians of T are the vertices with maximum s -values. The authors of [3, 21] establish that the algorithm stabilizes in a polynomial number of steps. See also [3, 21, 20] for self-stabilizing algorithms solving other facility location problems on tree-networks.

In this note, we propose self-stabilizing median computation algorithms for two classes of plane graphs: partial rectangular grid, even squaregraphs. Our algorithms are based on the fact that such graphs isometrically embed into the Cartesian product of two trees [13, 14] and that the median in the initial graph G can be derived from the medians in two spanning trees of G closely related to the two tree-factors. Using the sense of direction in the grid, the algorithm computes the tree-factors and apply the algorithm of [3, 21] to compute the medians of both

spanning trees. This computation is performed anonymously. For an even squaregraphs trees G , the algorithm first computes a spanning tree of G using the self-stabilizing spanning tree algorithm of Afek, Kutten, and Yung [1]. This algorithm requires that the vertices of the network are endowed with unique identities. Then the algorithm “repairs” this spanning tree in order to produce the two tree-factors and their spanning trees relatives, to which the median computation algorithm of [3, 21] is applied. The algorithms have a round complexity of $O(n)$ and $O(n^2)$, respectively. By self-stabilization, our algorithms are resilient to transient failures. Concerning non-transient failures like the permanent crash of a link or a processor, if the resulting topology is still a partial grid or an even squaregraph, then the algorithm will dynamically adjust to the changes. If not, and the crash creates a hole in the partial grid, then it is of course likely that our algorithm will not find the correct medians or even will not converge, because the computed factors could then contain cycles. Our algorithms have a modular structure because they consist of fair compositions of algorithms performing several basic tasks. As a consequence, any complexity improvement of one of these tasks would result into a better complexity of our algorithms.

The article is organized as follows. In the first part, we investigate the properties of partial grids, squaregraphs and their medians which are used in the algorithm. We also show that the characterization of medians of trees given by Gerstel and Zaks [41] extends to all cube-free median graphs. In the second part, we describe the algorithms for computing the medians and give a proof of the correctness as well as an upper bound on the time and round complexities. These are, to our knowledge, the first self-stabilizing algorithms for location problems on non-tree networks.

2 Partial grids, squaregraphs and their medians

2.1 Preliminaries

In a graph $G = (V, E)$, the *length* of a path from a vertex x to a vertex y is the number of edges in the path. The *distance* $d_G(x, y)$ (or $d(x, y)$ if G is obvious from the context) between x and y is the length of a shortest path connecting x and y . The interval $I(u, v)$ between two vertices u, v of G is the set $I(u, v) = \{x \in V : d(u, v) = d(u, x) + d(x, v)\}$. A subset $S \subseteq V$ is called *gated* [44, 37] if for each $v \notin S$ there exists a unique vertex $v' \in S$ (the gate of v in S) such that $v' \in I(v, u)$ for every $u \in S$. For every edge uv of G , define $W(u, v) = \{x \in V : d(u, x) < d(v, x)\}$. An *isometric subgraph* H of G is a subgraph of G such that $d_H(x, y) = d_G(x, y)$ for any pair of vertices x, y of H . Given two connected graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$, we say that G admits an *isometric embedding* into H if there exists a *mapping* $\alpha : V(G) \rightarrow V(H)$ such that $d_H(\alpha(x), \alpha(y)) = d_G(x, y)$ for all vertices $x, y \in V(G)$. The *Cartesian product* $H = H_1 \times H_2$ of two connected graphs H_1, H_2 is defined upon the Cartesian product of the vertex sets of the corresponding graphs (called *factors*), i.e., $V(H) = \{u = (u_1, u_2) : u_1 \in V(H_1), u_2 \in V(H_2)\}$. Two vertices $u = (u_1, u_2)$ and $v = (v_1, v_2)$ are adjacent in H if and only if the vectors u and v coincide except at one position i , in which we have two vertices u_i and v_i adjacent in H_i . The *distance* $d_H(x, y)$ between two

vertices $x = (x_1, x_2)$ and $y = (y_1, y_2)$ of H is $d_{H_1}(x_1, y_1) + d_{H_2}(x_2, y_2)$. Finally, recall that G is a *median graph* if for each triplet u, v, w of vertices the intersection $I(u, v) \cap I(v, w) \cap I(w, u)$ consists of a single vertex. Median graphs arise in several areas of discrete mathematics, geometry, and theoretical computer science (for a survey of properties of median graphs, see [12, 54, 60].)

2.2 Medians

In the following, we consider graphs with weighted vertices. A *weight function* is any mapping π from the vertex set to the positive real numbers. The total weighted distance of a vertex x in G is given by the *median function* $M_\pi(x) = \sum_u \pi(u)d(u, x)$. A vertex x minimizing this expression is a *median* (vertex) of G with respect to π , and the set of all medians is the *median set* $\text{Med}_\pi(G)$. For a subset of vertices $S \subseteq V$, denote by $\pi(S) = \sum_{s \in S} \pi(s)$ the weight of S . We continue with the following property of median functions:

Lemma 2.1 *For each edge uv in a graph G , $M_\pi(u) - M_\pi(v) = \pi(W(v, u)) - \pi(W(u, v))$.*

Proof. Indeed, since u and v are adjacent, we have $d(u, x) - d(v, x) = d(v, y) - d(u, y) = 1$ for any vertices $x \in W(v, u)$ and $y \in W(u, v)$. Since $d(x, z) = d(y, z)$ for any $z \notin W(u, v) \cup W(v, u)$, we conclude that $M_\pi(u) - M_\pi(v) = \sum_{x \in W(u, v)} \pi(x)(d(u, x) - d(v, x)) + \sum_{x \in W(v, u)} \pi(x)(d(u, x) - d(v, x)) = \pi(W(v, u)) - \pi(W(u, v))$. \square

Goldman and Witzgall [44] established that if the weight of a gated set S of G is larger than one half of the total weight, then $\text{Med}_\pi(G) \subseteq S$. In trees, in partial rectangular grids, in squaregraphs, and, more generally, in all median graphs, for each edge uv , the sets $W(u, v)$ and $W(v, u)$ are gated, and constitute a partition of G into two gated subgraphs [12, 54, 60]. We can then infer that these graphs satisfy the following *majority rule* (which is a folklore for trees):

Lemma 2.2 [10, 58] *If G is a median graph, then $u \in \text{Med}_\pi(G)$ if and only if $\pi(W(u, v)) \geq \pi(W(v, u))$ for each neighbor v of u . If T is a tree, then $\pi(W(u, v)) = \pi(W(v, u))$ if and only if $\text{Med}_\pi(T) = \{u, v\}$.*

It is well-known since C. Jordan (1869) that the median set of a tree consists of one or two adjacent vertices. For median graphs, and in particular for squaregraphs, the following generalization characterizes the structure of medians :

Lemma 2.3 [10, 58] *If G is a median graph, then $\text{Med}_\pi(G)$ induces a hypercube. In particular, if G is a squaregraph or a partial grid, then $\text{Med}_\pi(G)$ is a vertex, an edge, or a square.*

2.3 Partial grids and squaregraphs

A *partial rectangular grid* (a *polyomino* or a *rectangular system*) is the subgraph of the regular rectangular grid which is formed by the vertices and the edges of the grid lying either on a

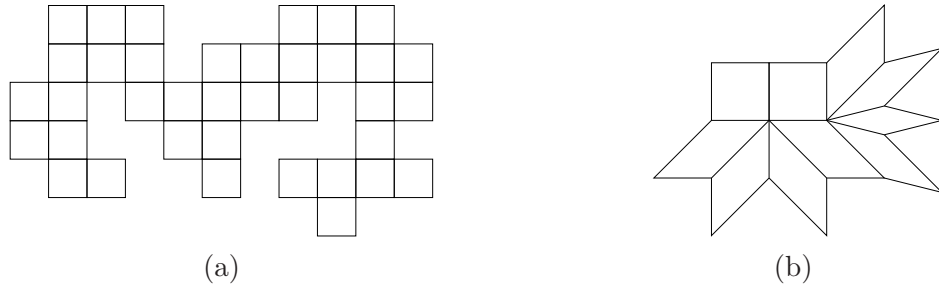


Figure 1: A rectangular grid (a) and an even squaregraph (b)

simple circuit of the grid (with possibly some vertices visited more than once) or inside the region bounded by this circuit. Every partial rectangular grid is a connected plane graph with inner faces of length four and inner vertices of degree four (the converse in general is not true). More generally, a *squaregraph* [13, 27] is a plane graph with inner faces of length four and inner vertices of degree at least four. An *even squaregraph* is a squaregraph in which all inner vertices have even degrees (see Figure 1). According to Lemma 2 of [13], squaregraphs constitute a particular subclass of median graphs.

The quadrangulations G arising in this paper can be viewed as special subgraphs of most popular network topologies (hypercubes, meshes, tori). Although extending the results of [3, 21] on trees to more general classes of graphs satisfying the majority rule seems to be an interesting problem in its own rights, this remark shows that our self-stabilizing algorithms for computing medians may be used to optimize the location of a shared resource in multi-user multi-processor systems. Indeed, many multi-processor systems (e.g., Intel iPSC860, Intel Paragon) may be configured as multi-user systems to better utilize the computational power [31, 39]. For this, processors are allocated to users so that no processor is simultaneously used by more than one user. As a result, the respective hypercube or mesh topology is divided into subnetworks specifying a restricted access to a portion of the network for particular users. From this perspective, our algorithms may be used to locate a shared resource in a multi-user systems with a hypercube, torus, or mesh topology in which the subgraph allocated to each user is a quadrangulation defined above. Notice that, motivated by this application, the broadcasting problem in submeshes and, in particular, in rectilinear cells has been considered in [39].

2.4 Isometric embedding into products of two trees

In this subsection, based on the results of [13, 14], we will describe the isometric embedding of partial rectangular grids and even squaregraphs $G = (V, E)$ into the Cartesian product of two trees.

First, let $G = (V, E)$ be a partial rectangular grid bounded by the cycle ∂G . Denote by E_1 the set of vertical edges of G and consider the graph $G_1 = (V, E_1)$. It is clear that the connected components of G_1 are paths of G with end-vertices on ∂G . Define the graph $T_1 = (V(T_1), E(T_1))$ whose vertices are the connected components of G_1 and two components

P' and P'' (i.e. vertices of T_1) are adjacent if and only if there exists an edge of G with one end in P' and another one in P'' . In the same way we can define the set E_2 of horizontal edges, the graph G_2 , and the tree $T_2 = (V(T_2), E(T_2))$. We obtain the following canonical embedding α of G into the Cartesian product $T_1 \times T_2$. For any vertex v of G , we set $\alpha_1(v)$ (resp. $\alpha_2(v)$) to be the connected component of v in G_1 (resp. G_2). The embedding is defined by $\alpha(v) = (\alpha_1(v), \alpha_2(v))$. It can be verified that α provides an isometric embedding of G into $T_1 \times T_2$. For all vertices x, y of G , we have $d_G(x, y) = d_{T_1}(\alpha_1(x), \alpha_1(y)) + d_{T_2}(\alpha_2(x), \alpha_2(y))$. From now on, we will identify a vertex v of G with the couple (P, Q) of vertical and horizontal paths to which v belongs. We will call such paths *fibers* because P is equal to the subgraph induced by all vertices v of G such that $\alpha_1(v) = P$ and Q is equal to the subgraph induced by all vertices v of G such that $\alpha_2(v) = Q$.

This canonical embedding of partial grids can be generalized to all graphs isometrically embeddable into Cartesian products of two trees. Such graphs are called *partial double trees* [14] and can be characterized in the following way:

Theorem 2.4 [14] *For a connected graph G the following conditions are equivalent:*

- (i) G is a partial double tree;
- (ii) G is a cube-free median graph without odd bipartite wheels;
- (iii) G is a median graph such that $\text{Link}(G)$ is bipartite;
- (iv) G is a median graph such that $\text{Link}(x)$ is bipartite for all vertices x of G .

The *link* of a vertex x in G is the graph $\text{Link}(v)$ whose vertices are the edges of G incident to v and where two edges are adjacent if and only if they are contained in a common 4-cycle. The *link graph* $\text{Link}(G)$ of a median graph G is then the union of the graphs $\text{Link}(v)$ for all vertices v of G . The condition (iii) of Theorem 2.4 can be equivalently reformulated in the following way:

(iii)' *the edges of G can be colored into two colors such that any 4-cycle of G is dichromatic with opposite edges having the same color.*

Let E_1 and E_2 be the edges of color 1 and color 2, respectively, in a coloring of G satisfying the condition (iii)'. Analogously to the case of partial grids, the connected components of the graphs $G_1 = (V, E_1)$ (resp. $G_2 = (V, E_2)$) are called the *fibers* of G_1 (resp. G_2). They are (gated) trees. Define the graphs $T_i = (V(T_i), E(T_i)), i = 1, 2$, whose vertices are the fibers of G_i and two fibers F' and F'' are adjacent if and only if there exists an edge of G with one end in F' and another one in F'' (see Figure 2). Equivalently, T_1 is obtained from G by collapsing all edges of E_2 and T_2 is obtained from G by collapsing all edges of E_1 . Then T_1 and T_2 are trees. It is shown in [14] that there exists an isometric embedding α of G into the Cartesian product $T_1 \times T_2$, so that for any vertex v of G , $\alpha(v) = (\alpha_1(v), \alpha_2(v)) = (P, Q)$, where $\alpha_1(v) = P$ and $\alpha_2(v) = Q$ are the fibers of the graphs G_1 and G_2 sharing the vertex v .

From condition (ii) of Theorem 2.4 we infer that even squaregraphs admit isometric embeddings into Cartesian products of two trees (see also [13]). The link of any vertex v of an

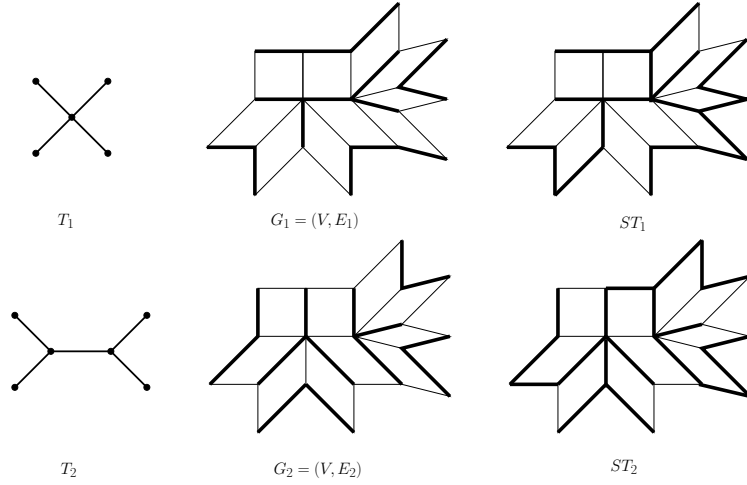


Figure 2: An even squaregraph and its trees T_i and ST_i

even squaregraph is either an even cycle (if v is an inner vertex of G) or a path (if v belongs to the boundary of G). Therefore each $\text{Link}(v)$ has exactly two bicolourings. To obtain a bicoloring of $\text{Link}(G)$, each vertex v of G must color its link in two colors in such a way that each edge uv is colored in the same color by u and by v . Equivalently, each vertex v must compute a bipartition $\{E_1^v, E_2^v\}$ of $\text{Link}(v)$ in a such a way that any edge uv of G belongs to E_1^u if and only if it belongs to E_1^v (we will show in Section 3.4 how this can be done in a self-stabilizing way). Then setting $E_1 := \bigcup\{E_1^v : v \in V\}$ and $E_2 := \bigcup\{E_2^v : v \in V\}$, we will obtain the desired bipartition of $\text{Link}(G)$. For even squaregraphs, the bipartition $\{E_1, E_2\}$ satisfies the following two conditions: (a) all “parallel” edges of G , i.e., edges which belong to the same equivalence class of the transitive closure of the binary relation “to be opposite edges of a common inner face of G ” all belong to the same color-class and (b) if we consider all edges incident to an inner vertex v of G , and number them counterclockwise, starting from an arbitrary edge, then the edges having numbers of the same parity all belong to the same color-class E_1 or E_2 . Note also that the set of all edges parallel to a given edge e (i.e. all edges that belong to the equivalence class of e) constitutes a cut-set of the graph G .

2.5 Isometric embedding into products of two trees and medians

The isometric embedding into the Cartesian product of two trees described above is used to establish one of the two properties on which our algorithm is based. Each vertex F_i of T_i is given the weight $\pi_i(F_i)$ equal to the total weight of vertices of G located in F_i .

Proposition 2.5 *Let G be an even squaregraph, a partial double tree, or a partial grid. A vertex u such that $\alpha(u) = (\alpha_1(u), \alpha_2(u))$ is a median vertex of G if and only if $F_1 = \alpha_1(u)$ and $F_2 = \alpha_2(u)$ are median vertices of the trees T_1 and T_2 endowed with the weight functions π_1 and π_2 respectively.*

Proof. Denote by F_i a fiber of the graph G_i which is a median vertex of the tree T_i , $i = 1, 2$. We assert that $F_1 \cap F_2 \neq \emptyset$. Suppose not and let $F_1 \cap F_2 = \emptyset$. From the definition of F_1 and F_2

we immediately conclude that F_2 is completely contained in the same connected component of the graph $G \setminus F_1$ obtained from G by removing all vertices of F_1 . Thus each vertex x of F_2 has its image $\alpha_1(x)$ in the same connected component of $T_1 \setminus F_1$. Denote by F'_1 the neighbor fiber of F_1 in the subtree of $T_1 \setminus F_1$ which contains F_2 . From Lemma 2.2 and the fact that F_1 is a median in T_1 , we deduce that $\pi_1(W_{T_1}(F_1, F'_1)) \geq \frac{1}{2}\pi_1(V(T_1)) = \frac{1}{2}\pi(V)$. In the same way, defining F'_2 to be the neighbor of F_2 in the connected component of $T_2 \setminus F_2$ which contains F_1 , we obtain $\pi_2(W_{T_2}(F_2, F'_2)) \geq \frac{1}{2}\pi(V)$. The sets of vertices of G whose images are respectively in $W_{T_1}(F_1, F'_1)$ and $W_{T_2}(F_2, F'_2)$ are disjoint and do not entirely cover the graph G . Since $\pi(x) > 0$ for any vertex x of G , we obtain a contradiction with Lemma 2.2. Thus $F_1 \cap F_2 \neq \emptyset$, whence there exists a vertex u of G such that $\alpha_1(u) = F_1$ and $\alpha_2(u) = F_2$ are medians in T_1 and T_2 respectively. Thanks to the isometric embedding, we obtain

$$\begin{aligned} M_\pi(u) &= \sum_{x \in V} \pi(x) d_G(u, x) = \sum_{x \in V} \pi(x) d_{T_1}(F_1, \alpha_1(x)) + \sum_{x \in V} \pi(x) d_{T_2}(F_2, \alpha_2(x)) \\ &= \sum_{R \in V(T_1)} \pi_1(R) d_{T_1}(F_1, R) + \sum_{Q \in V(T_2)} \pi_2(Q) d_{T_2}(F_2, Q). \end{aligned}$$

Writing up a similar expression for any other vertex v of G and using the fact that F_1 and F_2 are medians of T_1 and T_2 , respectively, we conclude that $M_\pi(u) \leq M_\pi(v)$, thus u is a median vertex of G . Conversely, the previous equality also shows that any median vertex of $\text{Med}_\pi(G)$ can be expressed as the intersection of two median paths, one of T_1 and another of T_2 . \square

Before proving the second property of medians of partial grids, even squaregraphs, and, more generally, of partial double trees, we define two particular spanning trees ST_1 and ST_2 of such graphs G . ST_1 contains all edges of E_1 plus exactly one edge from E connecting each pair of incident fibers of the graph $G_1 = (V, E_1)$ (see Figure 2). The choice of this edge is arbitrary (we can also select an edge belonging to the bounding cycle of G). We call such extra-edges the *switch edges* of ST_1 and denote them by E'_1 . Clearly, since all fibers of G_1 are trees, the graph $ST_1 = (V, E_1 \cup E'_1)$ is indeed a spanning tree of G . Analogously, we define the spanning tree $ST_2 = (V, E_2 \cup E'_2)$.

Proposition 2.6 *A vertex F_i of T_i ($i = 1, 2$) is a median vertex with respect to the weight function π_i if and only if F_i contains a median vertex u of the tree ST_i with respect to the weight function π .*

Proof. First suppose that u is a median vertex of ST_i (i.e., $u \in \text{Med}_\pi(ST_i)$) and let F_i be the vertex of T_i such that $\alpha_i(u) = F_i$. In other words, F_i is the fiber of G_i containing u . Suppose that F_i is not a median of T_i (i.e., $F_i \notin \text{Med}_{\pi_i}(T_i)$). Lemma 2.2 yields that $M_{\pi_i}(F') < M_{\pi_i}(F_i)$ for some vertex F' of T_i adjacent to F_i . By Lemma 2.1 and the definition of T_i we conclude that $M_{\pi_i}(F_i) - M_{\pi_i}(F') = \pi(W(x', x)) - \pi(W(x, x')) > 0$, where $x'x$ is any edge running between F' and F_i . If u has a neighbor u' in F' and uu' is a switch edge, then it can easily be seen from the definition of ST_i that all vertices of $W(x', x)$ (this set is defined in G) are closer to x' than to x in ST_i . This implies $M_\pi(u) - M_\pi(u') \geq \pi(W(x', x)) - \pi(W(x, x')) > 0$, contrary to the assumption that u is a median of ST_i . On the other hand, if the switch between F_i

and F' is the edge $p'p$ with $p' \in F'$ and $p \in F_i$, and we denote by u' the neighbor of u on the unique path connecting u with p in the tree-fiber F_i , then again, in the tree ST_i all vertices of $W(p', p)$ are closer to u' than to u . Since $\pi(W(p', p)) > \frac{1}{2}\pi(V)$, Lemma 2.2 yields that u is not a median vertex of ST_i , a contradiction.

Conversely, suppose that F_i is a median vertex of the tree T_i . We assert that $F_i \cap \text{Med}_\pi(ST_i) \neq \emptyset$. Remove from T_i all edges incident to F_i and denote by S_1, \dots, S_k the resulting subtrees of T_i not containing F_i . Then Lemmas 2.1 and 2.2 imply that $\pi_i(S_j) \leq \frac{1}{2}\pi(V)$ for any subtree S_j . Now, if we pick the switch edge $x_j u_j$ running between S_j and F_i with $x_j \in S_j$ and $u_j \in F_i$, then from the definition of the spanning tree ST_i we infer that S_j coincides with the set of all vertices which are closer to x_j than to u_j in ST_i . The majority rule for trees implies that $M_\pi(u_j) \leq M_\pi(x_j)$ holds in ST_i . Now, the median function M_π on trees is convex [59]. Since $M_\pi(u_j) \leq M_\pi(x_j)$, this implies that $M_\pi(x_j) \leq M_\pi(y_j)$ for any vertex $y_j \in S_j \setminus \{x_j\}$. Since any vertex z outside F_i is located in some subtree S_j , we conclude that $M_\pi(u_j) \leq M_\pi(x_j) \leq M_\pi(z)$ holds in ST_i . This shows that indeed F_i must contain at least one median vertex of the tree ST_i . \square

We obtain the following corollary as a direct consequence of the two previous properties:

Corollary 2.7 $u \in \text{Med}_\pi(G)$ if and only if $\alpha_i(u) \cap \text{Med}_\pi(ST_i) \neq \emptyset$, for $i = 1, 2$.

An algorithmic consequence of Proposition 2.6 and Corollary 2.7 is that in order to find $\text{Med}_\pi(G)$ it suffices to compute the medians of ST_1 and ST_2 , both endowed with the original weight function π , then take the fibers that contain these medians and return the intersection of these fibers.

2.6 On Gerstel and Zaks characterization of medians of cube-free median graphs

Gerstel and Zaks [41] characterized the medians of trees in the following nice way. Given a set S of $2k$ vertices of a graph G , a *pairing* is a partition of S into k disjoint pairs $[a_i, b_i]$. For a pairing P , set $\Gamma_P(S) = \sum_{i=1}^k d(a_i, b_i)$. Define $\Gamma(S) = \max\{\Gamma_P(S) : P \text{ is a pairing of } S\}$ and call a pairing P satisfying $\Gamma_P(S) = \Gamma(S)$ *maximal*. Denote by $\Delta(S)$ the minimum value of the median function for the weight function $\pi(v) = 1$ if $v \in S$ and $\pi(v) = 0$ if $v \notin S$. It is noticed in [41] that the inequality $\Gamma(S) \leq \Delta(S)$ holds for any subset S of vertices of even size of an arbitrary graph G . Moreover, it is shown in [41] that if G is a tree, then $\Gamma(S) = \Delta(S)$ and for any median vertex u there exists a maximal pairing P of S such that the unique path between every pair $[a_i, b_i] \in P$ passes through u . The authors of [41] ask for what classes of graphs the equality $\Gamma(S) = \Delta(S)$ holds for all subsets S . We present here a partial answer to this question by characterizing the median graphs fulfilling this property.

First notice that if S is a subset consisting of 4 vertices of the 3-cube Q_3 located at distance 2 from each of other (see Figure 3), then $\Gamma_P(S) = 4$ for any pairing of S , while $M_\pi(x) = 6$ for any vertex x of Q_3 . Thus $\Gamma(S) < \Delta(S)$. This strict inequality remains true in all median graphs hosting 3-cubes as isometric subgraphs. The following result shows that the 3-cube is the unique forbidden subgraph for the equality $\Gamma(S) = \Delta(S)$.

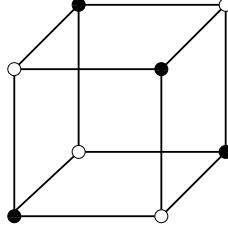


Figure 3: A subset S of Q_3 with $\Gamma(S) < \Delta(S)$

Proposition 2.8 $\Gamma(S) = \Delta(S)$ for any set S of even size $2k$ of a cube-free median graph G . Moreover, if $P = \{[a_i, b_i] : i = 1, \dots, n\}$ is a maximal pairing of S , then $\text{Med}_\pi(G) = \bigcap_{i=1}^k I(a_i, b_i)$, where $\pi(x) = 1$ if $x \in S$ and $\pi(x) = 0$ if $x \notin S$.

Proof. First we show that if $P = \{[a_i, b_i] : i = 1, \dots, n\}$ is a maximal pairing of S , then $\bigcap_{i=1}^k I(a_i, b_i) \neq \emptyset$. It is well-known [60] that the intervals of a median graph are convex and therefore are gated sets. Since the gated sets of a metric space satisfy the Helly property [37, 60], to show that $\bigcap_{i=1}^k I(a_i, b_i) \neq \emptyset$ it suffices to establish that the intervals defined by the maximal pairing P pairwise intersect. Suppose by way of contradiction that two such intervals $I(a_i, b_i)$ and $I(a_j, b_j)$ are disjoint. Then $I(a_i, b_i)$ and $I(a_j, b_j)$ can be separated by complementary halfspaces [60], more precisely, there exist two gated sets W', W'' of G such that $I(a_i, b_i) \subseteq W', I(a_j, b_j) \subseteq W'', W' \cap W'' = \emptyset$, and $W' \cup W'' = V$. Denote by T' the set of all vertices of W' having a neighbor in W'' . Then T' induces a gated subgraph of G [54, 60]. Since G is cube-free, T' is a tree. For any vertex v of G denote by v' its gate in T' . Notice that for any two vertices $v \in W'$ and $w \in W''$, there exists a shortest (v, w) -path passing via their gates v' and w' .

Consider the gates a'_i, b'_i, a'_j, b'_j in the gated tree T' of the vertices a_i, b_i, a_j, b_j , respectively. Now, it is well-known (and easy to prove) that any quadruplet of vertices of a tree has two different maximal pairings. At least one of them will match the gates of vertices from different pairs $[a_i, b_i], [a_j, b_j]$, say a'_i with a'_j and b'_i with b'_j . Let x be a common vertex of T' of the paths $I(a'_i, a'_j)$ and $I(b'_i, b'_j)$. Since there exists a shortest (a_i, a_j) -path passing via a'_i and a'_j , we conclude that $x \in I(a'_i, a'_j) \subseteq I(a_i, a_j)$. Analogously, $x \in I(b'_i, b'_j) \subseteq I(b_i, b_j)$. Hence

$$d(a_i, a_j) + d(b_i, b_j) = d(a_i, x) + d(x, a_j) + d(b_i, x) + d(x, b_j) \geq d(a_i, b_i) + d(a_j, b_j),$$

The last inequality (which is a consequence of the triangle inequality) must be strict, otherwise $x \in I(a_i, b_i) \cap I(a_j, b_j)$, contrary to the assumption that the intervals $I(a_i, b_i)$ and $I(a_j, b_j)$ are disjoint. Thus $d(a_i, a_j) + d(b_i, b_j) > d(a_i, b_i) + d(a_j, b_j)$. Consider the pairing P' obtained from P by setting $P' = (P - \{[a_i, b_i], [a_j, b_j]\}) \cup \{[a_i, a_j], [b_i, b_j]\}$. Clearly,

$$\Gamma_{P'}(S) = \Gamma_P(S) + d(a_i, a_j) + d(b_i, b_j) - d(a_i, b_i) - d(a_j, b_j) > \Gamma_P(S),$$

contrary to the choice of the pairing P . This contradiction shows that $\bigcap_{i=1}^k I(a_i, b_i) \neq \emptyset$.

Now, we will show that every vertex v of $\bigcap_{i=1}^k I(a_i, b_i)$ is a median vertex. Indeed, since

$v \in I(a_i, b_i)$ for all $i = 1, \dots, n$, we conclude that

$$M_\pi(v) = \sum_{i=1}^k (d(v, a_i) + d(v, b_i)) = \sum_{i=1}^k d(a_i, b_i) = \Gamma_P(S) = \Gamma(S) \leq \Delta(S),$$

whence $v \in \text{Med}_\pi(G)$.

It remains to show that $\text{Med}_\pi(G)$ is included in $\cap_{i=1}^k I(a_i, b_i)$. Suppose by way of contradiction that $\cap_{i=1}^k I(a_i, b_i)$ does not contain all median vertices. Since $\text{Med}_\pi(G)$ is a convex subgraph of G [58], we can select two adjacent median vertices u, u' such that $u \in \cap_{i=1}^k I(a_i, b_i)$ and $u' \notin \cap_{i=1}^k I(a_i, b_i)$. From Lemma 2.1 we conclude that the halfspaces $W(u, u')$ and $W(u', u)$ contain the same number of points of S . Therefore, if one of the halfspaces $W(u, u')$ and $W(u', u)$ contains both vertices of one pair of P , then the complementary halfspace must contain another such pair, contrary to the assumption that $\cap_{i=1}^k I(a_i, b_i)$ is non-empty. Thus for each pair $[a_i, b_i]$ of P exactly one vertex, say a_i , belongs to $W(u, u')$ and another vertex belongs to $W(u', u)$. Since $d(a_i, u') = d(a_i, u) + 1$, $d(b_i, u) = d(b_i, u') + 1$, and $u \in I(a_i, b_i)$, we immediately infer that $u' \in I(a_i, b_i)$ for all $[a_i, b_i] \in P$. This contradicts the choice of the vertex u' . Hence $\text{Med}_\pi(G) = \cap_{i=1}^k I(a_i, b_i)$, concluding the proof. \square

In [41], Gerstel and Zaks use the equality between $\Gamma(S)$ and $\Delta(S)$ to show that, given a network of tree topology, choosing a median vertex and routing all the information through this vertex is the best possible strategy, in terms of message complexity, for distributed sorting and ranking problems. For example, given a network $G = (V, E)$, a subset $S \subseteq V$, and an identifier for each processor of S , the distributed ranking problem consists of assigning (in a distributed manner) to each processor of S its rank in the sorted list of identifiers. Each processor executes a *protocol* that includes sending and receiving messages as well as local computations. The *message complexity* of a protocol is the maximum number of messages sent during any execution. The authors of [41] noticed that for any pairing P of S , it is possible to assign the identifiers in such a way that any distributed ranking algorithm will exchange at least $2\Gamma_P(S)$ messages. On the other hand, the protocol consisting in sending all identifiers to a median vertex, sorting them at this processor, and sending back the rank of each processor of S has message complexity $2\Delta(S)$. Since $2\Delta(S) = 2\Gamma(S)$ for trees, this is the best possible strategy for tree networks.

In view of Proposition 2.8, the results of [41] on distributed ranking and sorting problems extend from tree networks to cube-free median networks, in particular to partial rectangular grids, squaregraphs, and partial double trees .

3 Algorithms for the median problem

In the introduction, we outlined the self-stabilizing algorithm for the median problem in a tree proposed in [3, 21]. We continue with a more detailed account of the model used by this and our algorithms. Then we present the algorithm of [3, 21] and our algorithms for partial grids and even squaregraphs.

3.1 Computational model

The vertices of the graph $G = (V, E)$ are seen as processors executing the same algorithm. Each processor $v \in V$ has a memory whose value (its *state*) can be read by its neighbors, but can only be changed by v itself. A distributed algorithm is a set of rules (a pair of *precondition* and *command*) that describe how a processor has to change its current state (the command) according to the state of all its neighbors (the precondition or guard). We say that a rule R is *activable* at a processor v if the neighborhood of v satisfies the precondition of R . In this case, the vertex v is also said to be *activable*. If a rule R is activable in v , an *atomic move* for v consists of reading the states of all its neighbors, computing a new value of its state according to the command of R , and writing this value to the local memory. An *execution* is a sequence of moves. (Implicit) *termination* or *stabilization* is reached when there are no more activable rules. The described model is known as the interleaved (central daemon) asynchronous model of computation. This standard model for distributed computing has been introduced by Dijkstra [34] and used in many subsequent papers, in particular, in the papers [3, 21] presenting the self-stabilizing algorithms for medians of trees (in [21], the algorithms are expressed in the language of “guarded commands”). More recently, this model was extensively studied in [22]. In his thorough study of computational models, it is coined by Chalopin as the “interleaved cellular model” [25, Chapter 5]. In [1], it appears as the “local detection paradigm”.

A distributed algorithm is said to be *self-stabilizing* if an execution starting from any arbitrary global state has a suffix belonging to the set of legitimate states. For our purposes, we additionally suppose that the state variable of each vertex has a specific bit named the *median flag*. Then a global state is *legitimate* if the median flag of a vertex v is set up if and only if v is a median vertex of G . The *move complexity* of a self-stabilizing algorithm is the maximum number of moves that are performed until stabilization. A *round* [35] is a sequence of moves such that each vertex activable at the beginning of the round is activated at least once or becomes ineligible to perform its move. The *round complexity* of a self-stabilizing algorithm is the maximum number of rounds required by an execution to reach a legitimate state. Both our algorithms compute the median of partial grids and even squaregraphs for any weight function π , however for complexity issues, we assume that the weights of all vertices are polynomial in the size of the graph. This implies that $O(\log n)$ bits suffice to store the weights and the values of the median function computed by the algorithms. Notice also that in all three algorithms described below only $O(\log n)$ bits are used to store the state of each vertex.

We now specify the structural information that is used by each of the three median computation algorithms. In the algorithm for trees, neither vertices nor edges have identifiers. In this case, the system is said to be *anonymous*. Whereas in the algorithm for partial rectangular grids, vertices are anonymous but edges are not: for each vertex, there exists a port numbering (labeling of the outgoing edges) encoding the polar directions (north, west, etc...). This is used to allow the computation of the second neighborhood of each vertex in a self-stabilizing manner. Finally, in the algorithm for even squaregraphs, each processor has a

unique identifier and a numbering of ports that will be specified later. We point out only that using port numbers, when a vertex v tests its neighborhood state (to see whether there is an activable rule), then v has access to its own port numbers, but also, for any its neighbor u , to the port number that the vertex u associates to the edge uv .

3.2 Trees

Let $T = (V, E)$ be a tree with n vertices and let π be a weight function on V . We need the following notations:

- v .*s*-value is the local value of the vertex v . It is also called the s -value of v ;
- $\gamma_1(v) = \{u \in V : uv \in E\}$ is the set of neighbors of the vertex v in T ;
- $N_s(v) = \{u.s\text{-value} : u \in \gamma_1(v)\}$ is a multiset;
- $N_s^-(v) = N_s(v) \setminus \{\max(N_s(v))\}$.

The main result of [3, 21] is the proof of self-stabilization in polynomial time for the following algorithm (which we slightly modify to capture weighted medians as well). Algorithms are described by a list of rules, and to simplify the formulation of preconditions, we assume in the following that a rule is not activable if its execution would not change the state of the vertex.

MEDIAN TREE	
(v is a leaf)	\longrightarrow $v.s\text{-value} = \pi(v)$
(v is not a leaf)	\longrightarrow $v.s\text{-value} = \pi(v) + \sum(N_s^-(v))$.

In a stabilized state, the median vertices are those vertices whose s -value is greater than the s -values of all their neighbors. It is shown in [21] that MEDIAN TREE stabilizes in $O(e)$ rounds, where e is the maximum distance of a median to a leaf. Moreover, the algorithm makes $O(n^3 \cdot c_s)$ moves in the worst case, where c_s is the maximum initial s -value of any processor [21].

3.3 Partial rectangular grids

Let $G = (V, E)$ be a partial grid with n vertices bounded by the cycle ∂C . The first neighborhood $\gamma_1(v)$ of a vertex v of G is the set $V \cap \{v_N, v_S, v_E, v_W\}$, where v_N , v_S , v_E , and v_W are the vertices of the square grid located at the North, the South, the East, and the West of v , respectively. The second neighborhood $\gamma_2(v)$ of v in G is the set $V \cap \{v_{NE}, v_{NW}, v_{SE}, v_{SW}\}$, where v_{NE} , v_{NW} , v_{SE} and v_{SW} are the vertices of the square grid which are located at the North East, the North West, the South East, and the South West of v , respectively (see Figure 4). In the following, with $D \in \{N, S, E, W\}$, we use the notation $Neighb(v, D)$ if $v_D \in \gamma_1(v)$.

The model used by the algorithm MEDIAN PARTIAL GRID is similar to the one of MEDIAN TREE except that the system has the ‘‘polar’’ sense of direction (that is, the knowledge

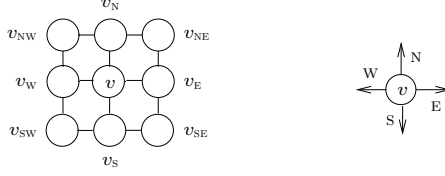


Figure 4: The first and second neighborhood of a vertex.

of the North, East, South and West outgoing edges). The algorithm MEDIANPARTIALGRID consists of three phases.

Phase 1. In this phase, in order to compute the sets $\gamma_{ST_1}(v)$ and $\gamma_{ST_2}(v)$ of its neighbors in the spanning trees ST_1 and ST_2 respectively, a processor v has to know about the existence of edges between its first neighborhood $\gamma_1(v)$ and its second neighborhood $\gamma_2(v)$. For example, for the tree ST_1 , this can be done by communicating with the first neighborhood $\gamma_1(v)$ and applying the following rule. For $D \in \{N, S, E, W\}$, $v_D \in \gamma_{ST_1}(v)$ if $Neighb(v, D)$ and

- either $D \in \{N, S\}$,
- or $D \in \{E, W\}$, and $\neg(Neighb(v, N) \wedge Neighb(v_N, D) \wedge Neighb(v_{ND}, S))$.

Phase 2. In this phase, each processor v runs the algorithm MEDIANTREE in parallel on each of the spanning trees ST_1 and ST_2 . The variable $v.s_i$ -value is the s -value of v for the tree ST_i . The weight of each vertex v of ST_i is its original weight $\pi(v)$ in the graph G . Then a median of ST_i can be identified by the fact that the s_i -value of the respective processor is maximum in its neighborhood on ST_i , $i = 1, 2$.

Phase 3. In this phase, a classical broadcasting self-stabilizing algorithm [35, chap.4, p 97] is replicated in both directions on every vertical (respectively, horizontal) path to compute two additional boolean variables $v.b_1$ and $v.b_2$. The variable $v.b_i$ will be set if the path $\alpha_i(v)$ is a median vertex of T_i .

The median vertices are finally obtained by remarking that, once the “vertical” and “horizontal” broadcasting algorithms stabilize, the median set of G is formed by all vertices v of the partial grid G for which $v.b_1 \wedge v.b_2$ is true.

The global algorithm MEDIANPARTIALGRID is a *fair composition* (as defined in Subsection 2.7 of [35]) of the algorithms described by the three phases. In a *fair execution* of the composite algorithm, every processor executes the steps of the composed algorithms in a round-robin way. The composite algorithm is self-stabilizing provided each of the involved algorithms is self-stabilizing and there is no circular dependency between them. More precisely, if we compose a sequence $\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_k$, of self-stabilizing algorithms, the algorithm \mathcal{A}_1 is used as a *server* algorithm for the *client* algorithm \mathcal{A}_2 . The composition of the algorithms \mathcal{A}_1 and \mathcal{A}_2 results in a self-stabilizing algorithm that is used as a server for the client \mathcal{A}_3 and so one. Let \mathcal{T}_i denote the task realized by the composition of the algorithms $\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_i$. In order to use the fair composition theorem of [35], we need to check that: (FC1) the variables used by

any algorithm \mathcal{A}_i are not modified by an algorithm \mathcal{A}_j for $j > i$; (FC2) for all $i = 1, \dots, k-1$, the algorithm \mathcal{A}_{i+1} is self-stabilizing for the task \mathcal{T}_{i+1} given the task \mathcal{T}_i . The condition (FC1) trivially holds for the algorithm MEDIANPARTIALGRID. The second condition (FC2) will be established in the proof of Theorem 3.1.

Theorem 3.1 *The algorithm MEDIANPARTIALGRID computes the median set of a partial grid G with n vertices in $O(n)$ rounds and $O(c_s n^5)$ moves.*

Proof. First we show that the algorithm stabilizes. The first phase is obviously self-stabilizing and produces two spanning trees. Given these trees, according to [3, 21], the second phase is self-stabilizing on each of them. Finally, the third phase is self-stabilizing by [35, chap.4, p 97]. This establishes the condition (FC2). Since the input-output binary dependency between the phases is obviously acyclic, the conditions of Theorem 2.2 of [35] are fulfilled, yielding that the algorithm MEDIANPARTIALGRID is self-stabilizing.

Notice further that the move complexity of Phase 1 is constant, while the move complexity of the broadcast is $O(h^2)$, where h is the maximal size of a vertical or a horizontal path of G . Taking into account the move complexity of MEDIANTREE [21], we conclude that the algorithm makes $O(c_s n^5)$ moves in the worst case, where c_s is the maximum initial s -value of any processor. Since our algorithm is obtained via a fair composition, its round complexity is the maximum of the round complexities of its phases [35]. In our case, this maximum is the round complexity $O(e)$ of MEDIANTREE, where $e \leq n$ is the maximum distance from a median to a leaf in the spanning tree. Hence the round complexity of MEDIANPARTIALGRID is $O(n)$.

Finally, we show the correctness of our algorithm, i.e. that in a global stabilized state each processor v that has its two boolean variables $v.b_1$ and $v.b_2$ set to true is a median of the partial grid G . Indeed, by Corollary 2.7 a vertex (processor) v is median in G if and only if it is on the vertical path of a median of ST_1 (thus variable $v.b_1$ set to true) and on the horizontal path of a median of ST_2 (variable $v.b_2$ set to true). Since the algorithm MEDIANTREE correctly computes the median set of each tree ST_1 and ST_2 , by Proposition 2.6 we conclude that our algorithm correctly computes the median of G . \square

Remark 1: In the assumption that the vertices of a tree have unique identities, Blair and Manne [20] presented a self-stabilizing algorithm for computing the median of a tree with better move complexity $O(n^2)$. They assert that the algorithm can be modified to not require unique identities, however the details of such a modification are not presented. We conjecture that instead of unique identities one can use polar port-numbering and adapt the proof of [20] to suit our needs. This would yield a total move complexity of $O(n^4)$ for MEDIANPARTIALGRID.

3.4 Even squaregraphs

Let $G = (V, E)$ be an even squaregraph (or, more generally, a partial double tree). The “irregular” structure of G does not allow an easy use of the sense of direction as in the case

of partial squaregrids. To obtain a bipartition of edges used in the construction of the trees T_1, T_2 and ST_1, ST_2 , we will use the self-stabilizing algorithm for constructing a spanning BFS-tree of a graph designed by Afek, Kutten, and Yung [1] (for a survey on other related algorithms for this problem, see [40]). This algorithm requires that each vertex v has a unique identifier $v.\text{id}$. This extra-information allows to break the symmetry in order to select as the root of the spanning tree the vertex having the highest identifier.

Our median self-stabilizing algorithm `MEDIANEVENSQUAREGRAPH` is a fair composition of four phases, which we briefly outline now. In the first phase, a BFS-tree of G is computed. In the second phase, using this BFS-tree, a bicoloring of $\text{Link}(G)$ is computed, i.e., a bipartition $\{E_1, E_2\}$ of edges of G such that two incident edges belonging to a common 4-cycle are included in different sets E_1 and E_2 . First, the root r of the tree chooses a bicoloring of $\text{Link}(r)$, then each neighbor v of r chooses a bicoloring of $\text{Link}(v)$ that agrees on the color of the edge vr with that of $\text{Link}(r)$. This process continues along the edges of the BFS-tree until each vertex v of G has chosen a bicoloring of $\text{Link}(v)$ that agrees on the color of the edge vv' with its father v' in the BFS-tree. In the proof of Theorem 3.2 given below, we show that this process ends up with the desired bicoloring $\{E_1, E_2\}$ of $\text{Link}(G)$. For this, we must show that if uv is an edge of G , then both vertices u and v have colored uv in the same color. This is obviously true if u is the father of v or if v is the father of u . Otherwise, we proceed by induction on the distance to the root r applying the following structural property of BFS-trees of cube-free median graphs established in Proposition 9.1 of [26] for a larger class of graphs:

Fellow Traveler Property: *If u and v are adjacent vertices of a cube-free median graph G , then their fathers u' and v' in any BFS-tree of G are adjacent in G .*

In Phase 3, the algorithm constructs the spanning trees $ST_i, i = 1, 2$. These spanning trees are obtained by “repairing” the BFS-tree computed during Phase 1. For computing ST_i , we keep the edge between a vertex v and its father u if vu belongs to E_i (i.e. u and v belong to the same fiber) or if v is closer to the root than all its neighbors in $G_i = (V, E_i)$. In this last case, the vertex v is the (unique) gate of r in the fiber that contains v and the edge vu will be the unique link between the fiber of v and the fiber of u . In the remaining case, we replace vu by an edge of E_i joining v to its closer to r neighbor in the fiber of v . With the trees ST_1 and ST_2 at hand, the last Phase 4 is similar to that for partial grids and reduces to the computation of the medians of the trees ST_1 and ST_2 . The correctness of this phase follows from Proposition 2.6. In each phase, we present the specific conditions which allow to test if the state of a vertex is legal or not for this phase. If the respective condition is not satisfied, then we describe the modifications which must be undertaken to move the system to a legal state. We continue now with a detailed description of the algorithm.

Phase 1. In this phase, we construct a spanning tree using the algorithm of Afek and al [1]. When this phase terminates, each vertex v has computed the identifier $v.\text{Root}$ of the root vertex of the resulting spanning BFS-tree, the identifier $v.\text{Parent}$ of its father in this tree and an integer $v.\text{Distance}$ which is the tree-distance between v and the root. This phase ends if the following condition holds in each vertex v :

Condition $st(v)$:

$$\{[(v.\text{Root} = v.\text{Id}) \wedge (v.\text{Parent} = v.\text{Id}) \wedge (v.\text{Distance} = 0)] \vee [(v.\text{Root} > v.\text{Id}) \wedge (v.\text{Parent} \in v.\text{Edge-list}) \wedge (v.\text{Root} = v.\text{Parent}.\text{Root}) \wedge (v.\text{Distance} = v.\text{Parent}.\text{Distance} + 1)]\} \wedge (v.\text{Root} \geq \max_{x \in v.\text{Edge-list}} x.\text{Root})$$

The algorithm, which is executed by each processor so that the system stabilizes in a state in which all these conditions are satisfied for each vertex, is described in details and analyzed in [1].

Phase 2. In this second phase of the algorithm, we aim to partition the edges of G into two subsets E_1 and E_2 so that two incident edges belonging to a common 4-cycle are included in different sets E_1 and E_2 . For a vertex v , we assume that the edges containing v have received *port numbers* $0, \dots, \text{deg}(v) - 1$ so that two edges incident to v and belonging to a common 4-cycle have numbers of different parity. For even squaregraphs, the 4-cycles are exactly the inner faces of G , therefore this condition can be achieved by assigning port numbers to the edges in the order in which they appear in a counterclockwise traversal around v . For each $v \in V$, the variable $v.\text{Color}$ equals i if all edges incident to v which have even port numbers belong to E_i and the remaining edges belong to E_{3-i} . Each vertex v runs the following algorithm:

$st(v) \wedge (v.\text{Id} = v.\text{Root})$	→	$v.\text{Color} := 1$
$st(v) \wedge (v.\text{Id} \neq v.\text{Root}) \wedge$ (the edge connecting the vertex v with $v.\text{Parent}$ occurs with the same parity in the adjacency list of v as in the adjacency list of $v.\text{Parent}$)	→	$v.\text{Color} := v.\text{Parent}.\text{Color}$
$st(v) \wedge (v.\text{Id} \neq v.\text{Root}) \wedge$ (the edge connecting the vertex v to $v.\text{Parent}$ occurs with different parities in the adjacency lists of v and $v.\text{Parent}$)	→	$v.\text{Color} := 3 - v.\text{Parent}.\text{Color}$

Note that the preconditions of second and third rules can be checked by v because it has access to its own port numbers as well as to the port number that each its neighbor u associates to the edge uv . We say that the condition $col(v)$ is satisfied by a vertex $v \in V$ if none of the rule is activable (recall that a rule is activable if its execution actually change the state of the vertex). Then Phase 2 terminates if the condition $col(v)$ is satisfied by all vertices $v \in V$ of G . At this time, from port-numbers and the values of the variable $v.\text{Color}$, each vertex v can decide for each its neighbor u if the edge uv belongs to E_1 or to E_2 . Denote by $v.\text{Edge-list}_i$ the list of neighbors of v in the graph $G_i = (V, E_i), i = 1, 2$. Note that these two lists are implicitly maintained because the membership of any neighbor u of v to one of these lists can be determined using the parity of the port number of the edge uv and the value of the variable $v.\text{Color}$. This ensures that only $O(\log n)$ bits are used to store the state of each vertex v .

Phase 3. In this phase, the algorithm constructs the spanning trees $ST_i, i = 1, 2$. For sake of simplicity, these trees will be rooted at the same vertex as the root of the spanning BFS-tree

computed in Phase 1. To encode the tree ST_i , we introduce for each $v \in V$ a new variable $v.\text{Parent}_i$ which will denote the father of the vertex v in the tree ST_i . The algorithm consists of “correcting” the spanning tree of Phase 1 by replacing $v.\text{Parent}$ with a vertex belonging to $v.\text{Edge-list}_i$ if this list contains a vertex located at the same distance to the root as the vertex $v.\text{Parent}$. Since all fibers of the graphs G_i are gated sets (in G), the tree obtained in this way has all the edges of E_i and exactly one switch edge of E_{3-i} running between each pair of neighboring fibers of G_i . Actually, if $v'v$ is a switch edge with $d_G(r, v') < d_G(r, v)$ and v belongs to the fiber F , then necessarily v is the gate of r in the fiber F and v' is the father of v in the BFS-tree. Indeed, the root r can be connected with any vertex u of F of G_i by a shortest path passing via the gate v of r in F , and thus the edge vv' will be included in the BFS-tree before the edge running from u to its father. Each processor $v \in V$ runs at this phase the following algorithm:

$col(v) \wedge (v.\text{Id} = v.\text{Root})$	\longrightarrow	$v.\text{Parent}_i := v.\text{Id}$
$col(v) \wedge (v.\text{Id} \neq v.\text{Root}) \wedge$ (the edge connecting v to $v.\text{Parent}$ belongs to E_i)	\longrightarrow	$v.\text{Parent}_i := v.\text{Parent}$
$col(v) \wedge (v.\text{Id} \neq v.\text{Root}) \wedge$ (the edge connecting the vertex v to $v.\text{Parent}$ belongs to E_{3-i}) \wedge ($v.\text{Parent}.\text{Distance} < \min_{x \in v.\text{Edge-list}_i} x.\text{Distance}$)	\longrightarrow	$v.\text{Parent}_i := v.\text{Parent}$
$col(v) \wedge (v.\text{Id} \neq v.\text{Root}) \wedge$ (the edge connecting the vertex v with $v.\text{Parent}$ belongs to E_{3-i}) \wedge ($v.\text{Parent}.\text{Distance} \geq \min_{x \in v.\text{Edge-list}_i} x.\text{Distance}$)	\longrightarrow	$v.\text{Parent}_i := \text{argmin}_{x \in v.\text{Edge-list}_i} x.\text{Distance}$

We say that the condition $st_i(v)$ is satisfied by the vertex v if none of the rule is activable. Since the spanning trees ST_1 and ST_2 are constructed at the same time, Phase 3 terminates if the condition $st_i(v)$ is satisfied at each vertex $v \in V$ and for each index $i \in \{1, 2\}$. At the end of this phase, each vertex $v \in V$ knows its father $v.\text{Parent}_i$ in the tree ST_i .

Phase 4. With trees ST_1 and ST_2 at hand, the algorithm is similar to that for partial grids. First, the algorithm MEDIAN TREE is used to compute the medians of the trees ST_1 and ST_2 . Then, using a self-stabilizing broadcasting algorithm, we set to “true” the variable $v.\text{b}_i$ of each vertex v belonging to the same connected component of the graph G_i as a median vertex of the tree ST_i . Once this broadcasting algorithm stabilizes, the median set $\text{Med}_\pi(G)$ of G is formed by all vertices v of G for which $v.\text{b}_1 \wedge v.\text{b}_2$ is true.

Theorem 3.2 *The algorithm MEDIANEVEN SQUAREGRAPH computes the median set of an even squaregraph G with n vertices in $O(n^2)$ rounds.*

Proof. The algorithm given by Afek et al. [1] for constructing a spanning tree stabilizes in $O(n^2)$ rounds. This shows that Phase 1 stabilizes in $O(n^2)$ rounds. The Phase 2 needs $O(n)$ rounds in the worst case. By induction we can show that when a vertex v located at distance i from the root is activated during round $i + 1$, the variable $w.\text{Color}$ of its father w , which is at distance $i - 1$ from the root, has already been correctly computed (by the induction hypothesis). Thus the rules of Phase 2 correctly compute the value of $v.\text{Color}$ using that of

w.Color. Since two edges incident to the same vertex v and belonging to the same 4-cycle have port numbers of different parity in the adjacency list of v , they will be included in different sets E_1 and E_2 by the algorithm. It remains to show that any edge uv is inserted in the same edge-list by both vertices u and v . For this we proceed by induction on $k := d_G(v, r) < d_G(u, r)$. Our previous argument shows that the assertion holds when v is the father of u in the BFS-tree. So, suppose that v' is the father of v and u' is the father of u in the BFS-tree and that $u' \neq v$. By the Fellow Traveler Property, since u and v are adjacent in G , their fathers u' and v' are also adjacent. Since $d_G(v', r) = k - 1$, by the induction hypothesis we conclude that u' and v' inserted the edge $u'v'$ in the same set, say E_1 . Now, since each vertex inserts two incident edges of a 4-cycle in different edge-lists and the edge connecting a vertex with its father is set in the same list by the two ends, we conclude that the edges $u'u$ and $v'v$ are in the lists E_2 of their extremities. This implies that the edge uv will be put in the list E_1 by both vertices u and v , thus establishing our assertion.

The rules of Phase 3 depend only of the information computed at Phases 1 and 2, thus in order to correctly compute the trees ST_1 and ST_2 in Phase 3, it suffices that each vertex is activated once in this phase. As to Phase 4, the algorithm MEDIAN TREE stabilizes in $O(e)$ rounds, where e is the largest eccentricity of a median vertex of G [21], thus its complexity is the same as that of broadcasting. Summarizing, we conclude that the construction in $O(n^2)$ rounds of a spanning tree of G dominates the overall complexity of our algorithm. We also showed that given that the task of each phase has stabilized, the next phase will also stabilize in a legitimate state. Thus, by Theorem 2.2 of [35], the composite algorithm MEDIANEVEN SQUAREGRAPH stabilizes as well. Finally, Proposition 2.6 establishes that MEDIANEVEN SQUAREGRAPH correctly computes the median set $\text{Med}(G)$. \square

Remark 2: In the assumption that for each vertex v of a partial double tree G a bipartition of $\text{Link}(v)$ is available, the four phases of the algorithm MEDIANEVEN SQUAREGRAPH can be easily adapted to compute the median of G (the correctness follows from Theorem 2.4, Proposition 2.6, and the Fellow Traveler Property). This partition could be easily computed locally if each vertex v can check whether two edges incident in v belong to a common 4-cycle. For instance, this condition is satisfied if, for any neighbor u of v , the vertex v can access the identities of all vertices adjacent to u .

Remark 3: Datta, Larmore, and Vemula [29] recently presented an optimal space self-stabilizing election algorithm for networks with unique identities. It uses $O(\log(n))$ bits per vertex and stabilizes in $O(n)$ rounds. This Election Algorithm induces a BFS tree that could be used in Phase 1 of our algorithm MEDIANEVEN SQUAREGRAPH. This would yield a round complexity of $O(n)$.

Note also that if in Phase 4 we use the algorithm of [20], as suggested in Remark 1, then this would require $O(\Delta \log(n))$ bits per vertex.

4 Conclusions and perspectives

In this paper, we presented two self-stabilizing algorithms for computing medians of two classes of plane graphs: partial rectangular grids and even squaregraphs. The algorithms are based on two facts: (i) our graphs are median graphs to which the majority rule of computing medians applies and (ii) our graphs G isometrically embed into the Cartesian product of two trees which can be found in a self-stabilizing way due to a bipartition of edges incident to each vertex. Using these properties, the median of G can be retrieved from the medians of two spanning trees of G closely related to the tree-factors. These are, to our knowledge, the first self-stabilizing algorithms for location problems on non-tree networks. We also show that the characterization of medians of trees given by Gerstel and Zaks [41] extends to all cube-free median graphs, a class of graphs that includes partial rectangular grids and even squaregraphs.

The following open questions can be formulated in relation with the results of our paper:

Question 1: Is it possible to design a self-stabilizing algorithm, which computes the medians of partial rectangular grids and even squaregraphs in a direct way (like the algorithms of [3, 21]), i.e., without the isometric embedding into the product of trees?

Our algorithms for computing the medians of partial rectangular grids (resp. of even squaregraphs) assume that we have a sense of direction (resp. that every vertex has a unique identifier). On the other hand, each processor uses only a constant number of $O(\log n)$ -bits variables. Moreover, we acknowledge that unique identities are quite costly but we underline that the second algorithm shows that the isometric embedding technique is very fruitful.

Our algorithm for computing the medians of even squaregraphs assume that every vertex has a unique identifier. In case of partial rectangular graphs, unique identifiers are not necessary since the polar sense of direction can be used to choose a unique switch edge between each pair of adjacent fibers and to compute the tree ST_1 and ST_2 . On the other hand, each processor uses only a constant number of $O(\log n)$ -bits variables. Moreover, we acknowledge that unique identities are quite costly but we underline that the second algorithm shows that the isometric embedding technique is very fruitful.

Question 2: Is it possible to design an algorithm not requiring these properties, e.g., assuming that all processors and links are anonymous? Additionally, is it possible to obtain such an algorithm if all processors are anonymous and each processor uses only $O(\log n)$ bits of memory?

One possible way to investigate such distributed computability is to follow the approach of [23] based on the concept of *fibration*. We conjecture that the median vertices are “fibration stable”. Nevertheless, this approach uses more than $O(\log n)$ bits of memory per processor.

Question 3: Design self-stabilizing algorithms for computing medians of median graphs, in particular, of general squaregraphs. Is it possible to find a median of a median graph G which can be isometrically embedded into the Cartesian product of k trees T_1, \dots, T_k using $O(k)$ variables per vertex?

It is shown in [58] (and more recently in [9]) that if a median graph $G = (V, E)$ is isometrically embedded into a hypercube Q , then for any weight function π , $\text{Med}_\pi(G)$ is the intersection of G with $\text{Med}_\pi(Q)$. Using the same proof as in Proposition 2.5 one can replace in this result the hypercube Q (which is a Cartesian product of edges) by the Cartesian product of the trees $H = T_1 \times \dots \times T_k$. The equality $\text{Med}_\pi(G) = \text{Med}_\pi(H) \cap V$ is the first step toward answering Question 3. Finally, notice that the squaregraphs isometrically embed into the Cartesian product of at most five trees [13].

Question 4: Rephrasing the first open problem of Gerstel and Zaks [41], characterize the graphs for which $\Gamma(S) = \Delta(S)$ for all subsets S of even size.

A *central vertex* of a graph G is a vertex v minimizing the largest distance from v to any other vertex in G . The *center* of G is the set of all central vertices. Bruell et al. [21] also designed a simple and nice self-stabilizing algorithm for computing the center of a tree. Unfortunately, there is no relationship between the center of a partial grid and the centers of two tree factors. In fact, it is possible to construct a partial grid whose central vertices are as far as we want from the intersection of the two paths corresponding to the central vertices of the tree factors.

Question 5: Design self-stabilizing algorithms for computing the centers of classes of graphs containing cycles, in particular, the centers of squaregraphs and kinggraphs (for the definition, see [27]).

Acknowledgments. We would like to acknowledge the two anonymous referees for a careful reading of the first version of the manuscript and many helpful suggestions.

References

- [1] Afek Y., Kutten S., and Yung M., The local detection paradigm and its applications to self-stabilization, *Theor. Comput. Sci.* **186** (1997) 199–229.
- [2] Aggarwal S., Kutten S., Time optimal self-stabilizing spanning tree algorithm, In *FSTTCS'93*, Springer LNCS, vol. 761, pp. 400–410, 1993.
- [3] Antonoiu G., Srimani P.K., A self-stabilizing distributed algorithm to find the median of a tree graph, *J. Comput. Syst. Sci.* **58** (1999) 215–221.
- [4] Antonoiu G., Srimani P.K., A self-stabilizing leader election algorithm for tree graphs, *J. Parallel Distrib. Comput.* **34** (1996) 227–232.
- [5] Antonoiu G., Srimani P.K., Distributed self-stabilizing algorithm for minimum spanning tree construction, In *Euro-Par'97*, Springer LNCS, vol. 1300, pp. 480–487, 1997.
- [6] Antonoiu G., Srimani P.K., Self-stabilizing protocol for mutual exclusion among neighboring nodes in a tree structured distributed system, *Parallel Alg. Appl.* **14** (1999) 1–18.

- [7] Arora A., Dolev S., Gouda M.G., Maintaining digital clocks in step, *Parallel Proc. Let.* **1** (1991) 11–18.
- [8] Awerbuch B., Kutten S., Mansour Y., Patt-Shamir B., Varghese G., Time optimal self-stabilizing synchronization, In *STOC'93*, ACM, pp. 652–661, 1993.
- [9] Balakrishnan K., Brešar B., Changat M., Klavzar S., Kovše M., Subhamathi A.R., Computing median and antimedian sets in median graphs, *Algorithmica* (to appear).
- [10] Bandelt H.-J., Barthélemy J.-P., Medians in median graphs, *Discr. Appl. Math.* **8** (1984) 131-142.
- [11] Bandelt H.-J., Chepoi V., Graphs with connected medians, *SIAM J. Discr. Math.* **15** (2002) 268-282.
- [12] Bandelt H.-J., Chepoi V., Metric graph theory and geometry: a survey, in: J. E. Goodman, J. Pach, R. Pollack (Eds.), *Surveys on Discrete and Computational Geometry. Twenty Years later*, *Contemp. Math.*, vol. 453, AMS, Providence, RI, 2008, pp. 49–86.
- [13] Bandelt H.-J., Chepoi V., Eppstein D., Combinatorics and geometry of finite and infinite squaregraphs, Electronic preprint arxiv:0905.4537, 2009.
- [14] Bandelt H.-J., Chepoi V., Eppstein D., Ramified rectilinear polygons: coordinatization by dendrons (in preparation).
- [15] Barcucci E., Del Lungo A., Nivat M., Pinzani R., Medians of polyominoes: a property for reconstruction, *Intern. Int. J. Imaging Syst. Technol.* **9** (1998) 69–77.
- [16] Barthélemy J.P., Janowitz M.F., A formal theory of consensus, *SIAM J. Discr. Math.* **4** (1991) 305-322.
- [17] Barthélemy J.P., Leclerc B., Monjardet B., On the use of ordered sets in problems of comparison and consensus of classifications, *J. Classification* **3** (1986) 187-224.
- [18] Barthélemy J.P., Monjardet B., The median procedure in cluster analysis and social choice theory, *Math. Soc. Sci.* **1** (1981) 235-268.
- [19] Buckley F., Harary F., *Distances in Graphs*, Redwood City, CA: Addison-Wesley, 1990.
- [20] Blair J.R.S, Manne F., Efficient self-stabilizing algorithms for tree networks, In *ICDCS'03*, IEEE Computer Society, pp 20-26, 2003.
- [21] Bruell S.B., Ghosh S., Karaata M.H., Pemmaraju S.V., Self-stabilizing algorithms for finding centers and medians of trees, *SIAM J. Comput.* **29** (1999) 600–614.
- [22] Boldi P., Vigna S., An effective characterization of computability in anonymous networks, In *DISC 2001*, Springer LNCS, vol. 2180, pp. 33–47, 2001.

- [23] Boldi P., Vigna S., Universal dynamic synchronous self-stabilization. *Distrib. Comput.*, **15** (2002) 137–153.
- [24] Chen N.S., Yu H.P., Huang S.T., A self-stabilizing algorithm for constructing spanning trees, *Inf. Proc. Let.* **39** (1991) 147–151.
- [25] Chalopin J., Algorithmique Distribuée, Calculs Locaux et Homomorphismes de Graphes, PhD Thesis Université Bordeaux I (2006).
- [26] Chepoi V., Graphs of some CAT(0) complexes, *Adv. Appl. Math.* **24** (2000) 125–179.
- [27] Chepoi V., Dragan F., Vaxès Y., Center and diameter problem in planar quadrangulations and triangulations, In *SODA'02*, ACM/SIAM, pp. 346–355, 2002.
- [28] Chepoi V., Fanciullini C., Vaxès Y., Median problem in some plane triangulations and quadrangulations, *Comput. Geom.* **27** (2004) 193–210.
- [29] Datta A., Larmore L., Vemula P., Self-Stabilizing Leader Election in Optimal Space. In *SSS 2008*. Springer LNCS, vol. 5340, pp 109-123
- [30] Daurat A., Del Lungo A., Nivat M., Medians of discrete sets according to a linear distance, *Discr. Comput. Geom.* **23** (2000), 465–483.
- [31] Das Sharma D., Pradhan D., Submesh allocation in mesh multicomputers using busy-lists: a best-fit approach with complete recognition capability, *J. Parallel and Distrib. Comput.* **36** (1996) 106-118.
- [32] Datta A.K., Derby J.L., Lawrence J.E., Tixeuil S., Stabilizing hierarchical routing, *J. Interconnexion Networks* **1** (2000) 283–302.
- [33] Delaët S., Nguyen D., Tixeuil S., Stabilité et auto-stabilisation du routage inter-domaine dans internet, In *RIVF*, Editions Suger Paris, pp. 139–144, 2003.
- [34] Dijkstra E. W., Self-stabilizing systems in spite of distributed control, *Comm. ACM* **17** (1974) 643–644.
- [35] Dolev S., *Self-stabilization*, MIT Press, 2000.
- [36] Dolev S., Israeli A., Moran S., Self-stabilization of dynamic systems assuming only read/write atomicity, In *PODC'90*, ACM, pp. 103–118, 1990.
- [37] Dress A., Scharlau R., Gated sets in metric spaces, *Aequat. Math.* **34** (1987) 112–120.
- [38] Flocchini P., Mans B., Santoro N., Sense of direction: definitions, properties and classes, *Networks* **32** (1998) 165–180.
- [39] Fraigniaud P., Vial S., One-to-all and all-to-all communications in partial meshes, *Parallel Proc. Let.* **9** (1999) 9-20.

- [40] Gärtner F., A survey of self-stabilizing spanning-tree construction algorithms, Technical Report IC/2003/38, Swiss Federal Institute of Technology, (2003).
- [41] Gerstel O., Zaks S., A new characterization of tree medians with applications to distributed algorithms, *Networks* **24** (1994) 23-29.
- [42] Ghosh S., Gupta A., Pemmaraju S.V., A self-stabilizing algorithm for the maximum flow problem, *Distrib. Comput.* **10** (1997) 167-180.
- [43] Ghosh S., Karaata M.H., A self-stabilizing algorithm for coloring planar graphs, *Distrib. Comput.* **7** (1993) 55-59.
- [44] Goldman A.J., Witzgall C.J., A localization theorem for optimal facility placement, *Transp. Sci.* **4** (1970) 406-409.
- [45] Gouda M.G., Herman T., Stabilizing unison, *Inf. Proc. Let.* **35** (1990) 171-175.
- [46] Gouda M.G., Schneider M., Maximum flow routing, In *WSS'95*, pp. 1-13, 1995.
- [47] Hedetniemi S.T., Jacobs D.P., Srimani P.K., Maximal matching stabilizes in time $o(m)$, *Inf. Proc. Let.* **80** (2001) 221-223.
- [48] Hendry G.R.T, On graphs with prescribed medians I, *J. Graph Th.* **9** (1985) 477-481.
- [49] Herman T., Ghosh S., Stabilizing phase-clocks, *Inf. Proc. Let.* **54** (1995) 259-265.
- [50] Huang T.C., Lin J., Chen H., A self-stabilizing algorithm which finds a 2-center of a tree, *Comp. Math. Appl.* **40** (2000) 607-624.
- [51] Kariv O., Hakimi S.L., An algorithmic approach to network location problems, I,II, *SIAM J. Appl. Math.* **37** (1979) 513-538, 539-560.
- [52] Korach E., Rotem D., Santoro N., Distributed algorithms for finding centers and medians in networks, *ACM Trans. Program. Lang. Syst.* **6** (1984) 380-401.
- [53] Métivier Y., N. Saheb, Medians and centers of polyominoes, *Inf. Proc. Let.* **57** (1981) 175-181.
- [54] Mulder H.M., *The Interval Function of a Graph*, Math. Centre Tracts 132, Amsterdam, 1980.
- [55] Nesterenko M., Mizuno M., A quorum-based self-stabilizing distributed mutual exclusion algorithm, *J. Parallel Distrib. Comput.* **62** (2002) 284-305.
- [56] Santoro N., Sense of direction, topological awareness, and communication complexity, *ACM SIGACT News* **16** (1984) 50-56.
- [57] Shukla S., Rosenkrantz D., Ravi S., Developing self-stabilizing coloring algorithms via systematic randomization, In *Proc. Int. Workshop Parallel Proc.*, Tata McGraw-Hill, pp 668-673, 1994.

- [58] Soltan P., Chepoi V., The solution of the Weber problem for discrete median metric spaces (in Russian), Trudy Tbilisskogo Mat. Inst. **85** (1987) 52–76 (English translation: in Topological, Projective and Combinatorial Properties of Spaces, Chogoshvili G.S. ed., Nova Science, pp. 89–129, 1992.)
- [59] Tansel B.C, Francis R.L, Lowe T.J., Location on networks: a survey, Management Sci. **29** (1983) 482–511.
- [60] van de Vel M.L.J., Theory of Convex Structures, North-Holland, Amsterdam, 1993.