A self-stabilizing algorithm for the median problem in partial rectangular grids and their relatives[1]

Victor Chepoi, Tristan Fevat, Emmanuel Godard, and Yann Vaxès

LIF-Laboratoire d'Informatique Fondamentale de Marseille,
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 in finding the set of all median vertices of G. In this note, we present a self-stabilizing algorithm for the median problem in partial rectangular grids. Our algorithm is 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 in finding the set of all median vertices. The median problem arises with one of the basic models in discrete facility location [50] and with majority consensus in classification and data analysis [9, 15]. This is also a classical topic in graph theory [16, 50]. 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 [24], and a few other classes of graphs. A distributed algorithm for computing medians in graphs is given in [44]. Finally notice that some papers [9, 49] investigate the structural properties of median sets in special classes of graphs.

[1]An 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"(TAGADA project).

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 [33] 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 [28]. 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 [27]. 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 [25, 26], synchronization [7, 8, 41], leader election [4, 29], spanning tree construction [2, 5], maximum flow and maximum matching [34, 39], graph coloring [35], and mutual exclusion in several kinds of networks [46, 6].

Antonoiu, Srimani [3] and Bruell et al. [18] proposed a strikingly simple and nice self-stabilizing algorithm for computing the median set of a tree T. The state of each node 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 happens to be a "valid" global state as the medians of T are the vertices with maximum s-values. The authors of [3, 18] establish that the algorithm stabilizes in a polynomial number of steps. See also [3, 18, 17] 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 and even squaregraphs. Our algorithms are based on the fact that such graphs isometrically embed into the Cartesian product of two trees 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, 18] to compute the medians of both spanning trees. This computation is performed anonymously. For an even squaregraph 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 needs unique identities to be available for every node of the network. 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, 18] 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.

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 [33] 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* [36, 30] 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 $W(u, v) = \{x \in V : d(u, x) < d(v, x)\}$. 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* $\alpha : V(G) \to V(H)$ 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 $d_{H_1}(x_1, y_1) + d_{H_2}(x_2, y_2)$.

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 $M_\pi(x) = \sum_u \pi(u)d(u, x)$. 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* $\mathrm{Med}_\pi(G)$. 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 $M_\pi(u) - M_\pi(v) = \sum_{x \in W(u,v)} \pi(x)(d(u,x) - d(v,x)) + \sum_{x \in W(v,u)} \pi(x)(d(u,x) - d(v,x)) = \pi(W(v,u)) - \pi(W(u,v))$. □

Goldman and Witzgall [36] established that if the weight of a gated set S of G is larger than one half of the total weight, then $\mathrm{Med}_\pi(G) \subseteq S$. 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. Recall that G is a *median graph* if for each triplet u, v, w the intersection $I(u,v) \cap I(v,w) \cap I(w,u)$ consists of a single vertex. Recall that G is a *median graph* if for each triplet u, v, w 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 [11, 45, 51]. One of the basic properties of median graphs is that for any edge uv the sets $W(u,v)$ and $W(v,u)$ constitute a partition of G into two gated subgraphs. We can then infer that these graphs satisfy the following *majority rule* (which is a folklore for trees):

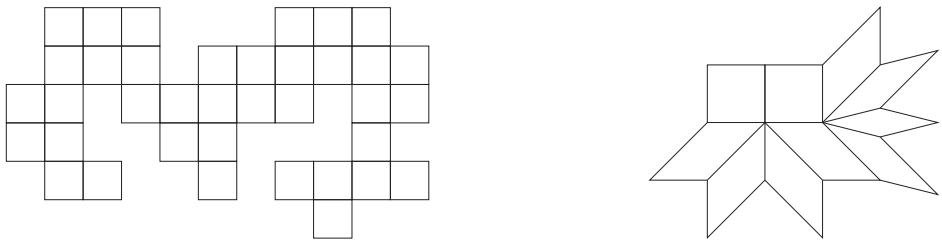
Lemma 2.2 *[9, 49] If G is a median graph, then $u \in Med_\pi(G)$ iff $\pi(W(u,v)) \geq \pi(W(v,u))$ for each neighbor v of u. If T is a tree, then $\pi(W(u,v)) = \pi(W(v,u))$ if and only if $Med_\pi(T) = \{u,v\}$.*

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 *[9, 49] If G is a median graph, then $Med_\pi(G)$ induces a hypercube. In particular, if G is a squaregraph or a partial grid, then $Med_\pi(G)$ is a vertex, an edge, or a square.*

2.3 Partial grids and squaregraphs

A *rectangular system* or a *partial rectangular grid* is the subgraph of the regular rectangular grid which is formed by the vertices and the edges of the grid lying either on a 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* [23] 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 Fig. 1). Squaregraphs constitute a particular subclass of median graphs.



(a) (b)

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

2.4 Isometric embedding into products of two trees

Now, we will describe the isometric embedding of partial rectangular grids and even squaregraphs $G = (V, E)$ into the Cartesian product of two trees. For this we will use the notations in [12, 23]. For a squaregraph or a partial grid G denote by ∂G the bounding cycle of G.

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'' 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 of G with the couple of vertical and horizontal paths to which it belongs. We call such a path P a *fiber*, as P is equal to the subgraph induced by $\alpha_1^{-1}(P)$.

This canonical embedding of partial grids can be generalized to all graphs isometrically embeddable into Cartesian products of two trees. It was established in [12] that a graph G can be embedded into the Cartesian product of two trees if and only if G is a cube-free median graph without odd bipartite wheels. In particular, from this characterization follows that even squaregraphs admit such embeddings. To derive the embedding, the edges of an even squaregraph $G = (V, E)$ are divided into two sets E_1 and E_2 subject to the constraint that two incident edges e_1 and e_2 of a common inner face of G belong to different edge-sets. Equivalently, if we define the *side-graph* of G as the graph having the edges of G as the vertex-set and two edges e_1, e_2 of G are adjacent in the side-graph if and only if e_1 and e_2 are incident sides of some inner face of G, then the side-graph is bipartite. Note that the bipartition $\{E_1, E_2\}$ of E satisfies the following two conditions: (i) 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 (ii) if we consider all edges incident to an inner vertex v of G, and number them

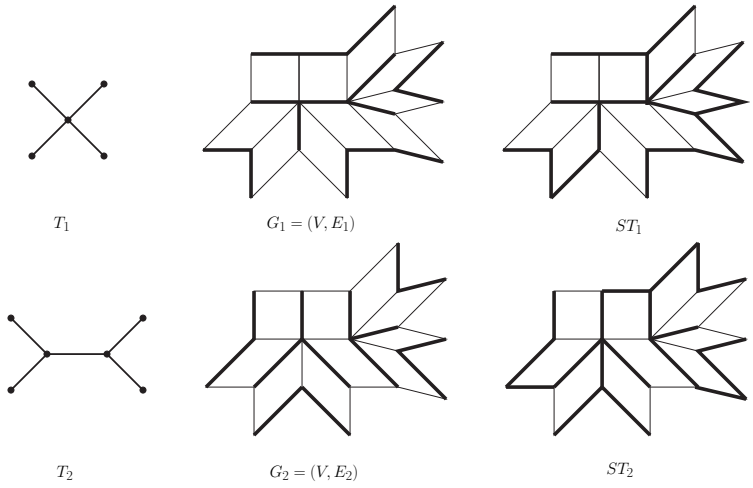


Figure 2: An even squaregraph and its trees T_i and ST_i

counterclockwise, starting with 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.

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''. Then T_1 and T_2 are trees. We obtain an isometric embedding α of G into the Cartesian product $T_1 \times T_2$ of the two trees, 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 [12].

This isometric embedding into the Cartesian product of two trees is used to establish one of the two properties on which our algorithms are 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 2.4 *Let G be an even squaregraph or a partial grid. A vertex $u = (\alpha_1(u), \alpha_2(u))$ 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 all vertices x of F_2 have their image $\alpha_1(x)$ in the same connected component of $T_1 \setminus F_1$. Denote by F_1' the neighbor 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 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 m of G such that $\alpha_1(m) = F_1$ and $\alpha_2(m) = F_2$ are medians in T_1 and T_2 respectively. Thanks to the isometric embedding, we obtain

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

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 $M_\pi(m) \leq M_\pi(v)$, thus m is a median vertex of G. Conversely, the previous equality also shows that any median vertex of $\mathrm{Med}_\pi(G)$ 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 and even squaregraphs, we define two particular spanning trees ST_1 and ST_2 of an even squaregraph G. ST_1 contains all edges of E_1 plus exactly one edge running between each pair of incident fibers of the graph $G_1 = (V, E_1)$. 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 2.5 *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 m of the tree ST_i with respect to the weight function π.*

Proof. First suppose that m is a median vertex of ST_i (i.e., $m \in \mathrm{Med}_\pi(ST_i)$) and let F_i be the vertex of T_i such that $\alpha_i(m) = F_i$. In other words, F_i is the fiber of G_i containing m. Suppose that F_i is not a median of T_i (i.e., $F_i \notin \mathrm{Med}_{\pi_i}(T_i)$). 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 m has a neighbor m' in F' and mm' 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 $M_\pi(m) - M_\pi(m') \geq \pi(W(x', x)) - \pi(W(x, x')) > 0$, contrary to the assumption that m 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 m' the neighbor of m on the unique path connecting m with p in the tree-fiber F_i, then again, in the tree ST_i all vertices of $W(p', p)$ are closer to m' than to m. Since $\pi(W(p', p)) > \frac{1}{2}\pi(V)$, Lemma 2.2 yields that m 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 $F_i \cap \mathrm{Med}_\pi(ST_i) \neq \emptyset$. Remove from T_i all edges incident to F_i and denote by $S_1, \ldots, S_k$ 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_jm_j running between S_j and F_i with $x_j \in S_j$ and $m_j \in F_i$, 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 m_j in ST_i. The majority rule for trees implies that $M_\pi(m_j) \leq M_\pi(x_j)$ holds in ST_i. Now, the median function M_π on trees is convex [50]. Since $M_\pi(m_j) \leq M_\pi(x_j)$, 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 $M_\pi(m_j) \leq M_\pi(x_j) \leq M_\pi(z)$ holds in ST_i. This shows that indeed F_i must contain at least one median vertex of the tree ST_i. □

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

Corollary 2.6 *$m \in Med_\pi(G)$ if and only if $\alpha_i(m) \cap Med_\pi(ST_i) \neq \emptyset$, for $i = 1, 2$.*

2.5 On Gerstel and Zaks characterization of medians

Gerstel and Zaks [33] characterized the medians of trees in the following nice way. Given a set S of $2n$ vertices of a graph G, a *pairing* is a partition of S into n disjoint pairs $[a_i, b_i]$. For a pairing P, set $\Gamma_P(S) = \sum_{i=1}^{n} d(a_i, b_i)$. 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 [33] 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 [33] that if G is a tree, then $\Gamma(S) = \Delta(S)$ and for any median vertex m there exists a maximal pairing P of S such that the unique path between every pair $[a_i, b_i] \in P$ passes through m. The authors of [33] 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 each of other at distance 2 (see Fig. 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 $\Gamma(S) = \Delta(S)$.

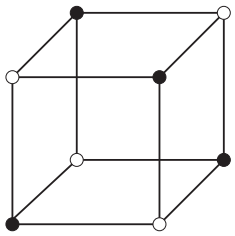


Figure 3: A subset S of Q_3 with $\Gamma(S) < \Delta(S)$

Proposition 2.7 *$\Gamma(S) = \Delta(S)$ for any set S of even size $2n$ of a cube-free median graph G. Moreover, if $P = \{[a_i, , b_i] : i = 1, \ldots, n\}$ is a maximal pairing of S, then $Med_\pi(G) = \cap_{i=1}^{n} I(a_i, b_i)$, where $\pi(x) = 1$ if $x \in S$ and $\pi(x) = 0$ if $x \notin S$.*

Proof. First we show that if $P = \{[a_i, b_i] : i = 1, \ldots, n\}$ is a maximal pairing of S, then $\cap_{i=1}^{n} I(a_i, b_i) \neq \emptyset$. It is well-known [51] 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 [30, 51], to show that $\cap_{i=1}^{n} I(a_i, b_i) \neq \emptyset$ 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 [51], 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 [45, 51]. 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 $u \in W'$ and $v \in W''$, there exists a shortest (u, v)-path passing via the vertices u' and v'.

Consider the gates a_i', b_i', a_j', b_j' in the gated tree T' of the vertices a_i, b_i, a_i, 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 $\cap_{i=1}^{n} I(a_i, b_i) \neq \emptyset$.

Now, we will show that every vertex v of $\cap_{i=1}^{n} I(a_i, b_i)$ is a median vertex. Indeed, since $v \in I(a_i, b_i)$ for all $i = 1, \ldots, n$, we conclude that

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

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

It remains to show that $\mathrm{Med}_\pi(G)$ is included in $\cap_{i=1}^{n} I(a_i, b_i)$. Suppose by way of contradiction that $\cap_{i=1}^{n} I(a_i, b_i)$ does not contain all median vertices. Since $\mathrm{Med}_\pi(G)$ is a convex subgraph of G [49], we can select two adjacent median vertices m, m' such that $m \in \cap_{i=1}^{n} I(a_i, b_i)$ and $m' \notin \cap_{i=1}^{n} I(a_i, b_i)$. From Lemma 2.1 we conclude that the halfspaces $W(m, m')$ and $W(m', m)$ contain the same number of points of S. Therefore, if one of the halfspaces $W(m, m')$ and $W(m', m)$ contains both vertices of one pair of P, then the complementary halfspace must contain another such pair, contrary to the assumption that $\cap_{i=1}^{n} I(a_i, b_i)$ is non-empty. Thus for each pair $[a_i, b_i]$ of P exactly one vertex, say a_i, belongs to $W(m, m')$ and another vertex

belongs to $W(m', m)$. Since $d(a_i, m') = d(a_i, m) + 1, d(b_i, m) = d(b_i, m') + 1$, and $m \in I(a_i, b_i)$, we immediately infer that $m' \in I(a_i, b_i)$ for all $[a_i, b_i] \in P$. This contradicts the choice of the vertex m'. Hence $\mathrm{Med}_\pi(G) = \cap_{i=1}^n I(a_i, b_i)$, concluding the proof. □

In [33], 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 node is the best possible strategy, in terms of message complexity, for distributed sorting and ranking problems in networks of tree topology. For example, in the case of distributed *ranking problem*, given a distributed network $G = (V, E)$ of processors and a subset $S \subset V$, each processor of S having an identifier, it is necessary to assign (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, receiving messages, and local computations. The *message complexity* of a protocol is the maximum number of messages sent during any execution. The authors of [33] 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 node, 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 our Proposition 2.7, the results of [33] on distributed ranking and sorting problems extend from tree networks to cube-free median networks, in particular to partial rectangular grids and squaregraphs.

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, 18]. We continue with a more detailed account of the model used by this and our algorithms. Then we present the algorithm of [3, 18] and our algorithms for partial grids and even squaregraphs.

3.1 Computational model

The nodes 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 node v is also said to be *activable*. If a rule R is activable in v, an *atomic move* for v consists in 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. This is indeed an interleaved (central deamon) asynchronous model of computation. (Implicit) termination or stabilization is reached when there are no more activable rules. The described model is a standard model for distributed computing

originally introduced by Dijkstra [27] and used in many following papers. In particular, it was extensively studied in [19]. It was used in [18], where the algorithms are expressed in the language of "guarded commands". In his thorough study of computational models, it is coined by Chalopin as the "interleaved cellular model"[21, 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 node has a specific bit named the *median flag*. Then a global state is *legitimate* if the median flag of a node v is set up if and only if v is a median vertex of G. The *time complexity* of a self-stabilizing algorithm is the maximum number of moves that are performed until stabilization. A *round* [28] is a sequence of moves such that each node activable at the beginning of the round is activated at least once. The *round complexity* of a self-stabilizing algorithm is the maximum number of rounds required by an execution to reach a legitimate state. Whenever we compose two or more self-stabilizing algorithms, then this will be done as a *fair composition* as defined in Subsection 2.7 of [28]. Additionally, the input-output binary dependency between our algorithms will be acyclic. Then Theorem 2.2 of [28] guarantees the self-stabilization of the resulting composite algorithm. In all three algorithms described below, only $O(\log n)$ bits are used to store the state of each node.

We specify now the structural information that is used by each of the three median computation algorithms. In the algorithm for trees, neither nodes nor edges have identifiers. In this case, the system is said to be *anonymous*. Whereas in the algorithm for partial rectangular grids, nodes are anonymous but edges are not: for each node, there exists a labeling of the outgoing edges that has the property of (weak) "sense of direction", allowing to compute the second neighborhood of each node in a self-stabilizing manner. Informally, a system represented by a directed graph is said to have *sense of direction* if it is possible to know, from the labels associated to the edges, whether different walks from any given node v end up in the same node or not. The use of sense of direction in a distributed system often leads to significant improvements on computability and complexity [31]. Finally, in the algorithm for even squaregraphs, each processor has a unique identifier (as a matter of fact, it is unclear for us whether this is a necessary structural information).

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:
- $N_s^-(v) = N_s(v) \backslash \{\max(N_s(v))\}$.

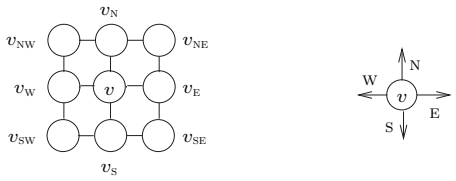


Figure 4: The first and second neighborhood of a vertex.

The main result of [3, 18] 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 all the following that a rule is not activable if its execution would not change the state of the node.

MEDIAN TREE		
(v is a leaf)	$\longrightarrow$	$v.s$-value$= \pi(v)$
(v is not a leaf)	$\longrightarrow$	$v.s$-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 [18] that MEDIANTREE 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 [18].

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 $V \cap \{v_{\rm N}, v_{\rm S}, v_{\rm E}, v_{\rm W}\}$, where $v_{\rm N}$, $v_{\rm S}$, $v_{\rm E}$ and $v_{\rm W}$ 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 $V \cap \{v_{\rm NE}, v_{\rm NW}, v_{\rm SE}, v_{\rm SW}\}$, where $v_{\rm NE}$, $v_{\rm NW}$, $v_{\rm SE}$ and $v_{\rm SW}$ 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 3). In the following, with $\mathrm{D} \in \{\mathrm{N}, \mathrm{S}, \mathrm{E}, \mathrm{W}\}$, we use the notation $Neighb(v, \mathrm{D})$ if $v_{\rm D} \in \gamma_1(v)$.

The model used by the algorithm MEDIANPARTIALGRID is similar to the one of MEDIANTREE except that the system has the "polar" sense of direction (that is, the knowledge 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 $\mathrm{D} \in \{\mathrm{N}, \mathrm{S}, \mathrm{E}, \mathrm{W}\}$, $v_{\rm D} \in \gamma_{ST_1}(v)$ if $Neighb(v, \mathrm{D})$ and

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

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. 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 [28, chap.4, p 97] is replicated in both directions on every vertical (respectively, horizontal) path to compute two additional boolean variables $v.\mathsf{b_1}$ and $v.\mathsf{b_2}$. The variable $v.\mathsf{b_i}$ will be set if the path $\alpha_i(v)$ is a median vertex of T_i.

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 $v.\mathsf{b_1} \wedge v.\mathsf{b_2}$ is true.

Theorem 3.1 *The algorithm* MedianPartialGrid *computes the median set of a partial grid G with n vertices in $O(n)$ rounds and $O(c_s n^3)$ moves.*

Proof. First we show that the algorithm stabilizes. Indeed, by simulating the execution of MedianTree on the two spanning trees ST_1 and ST_2, we obviously maintain the self-stabilization because the read and written variables are distinct (we use s_1 for the tree ST_1 and s_2 for ST_2). By composing these algorithms with the broadcasting algorithm, the self-stabilization is still maintained according to Theorem 2.2 of [28].

As to the time complexity, notice that 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. Since the time complexity of MedianTree is greater than the time complexity of classical self-stabilizing broadcasting algorithms, the time complexity of MedianPartialGrid is of the same order as the time complexity of MedianTree which is given in [18], concluding the proof. In the same way the round complexity of MedianTree is $O(e)$, where $e \leq n$ is the maximum distance from a median to a leaf in the spanning tree, yielding $O(n)$ round complexity for MedianPartialGrid.

Finally, we show the correctness of our algorithm, i.e. that in a global stabilized state the processors v that have their two boolean variables $v.b_1$ and $v.b_2$ set to true are medians of the partial grid G. Indeed, by Corollary 2.6 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, we are done. □

3.4 Even squaregraphs

Let $G = (V, E)$ be an even squaregraph. 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 [32]). This algorithm requires that each vertex v has a unique identifier $v.\mathsf{Id}$. 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 consists of four phases. In each phase, we present the specific conditions which allow to test if the state of a vertex is legal or not for the current phase. If the respective condition is not satisfied, then we describe the modifications which must be undertaken in order to return the system to a legal state. We establish that after a finite number of activations, the corresponding conditions of the current phase are satisfied by all vertices, and thus the next phase can start.

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 $v.\mathsf{Root}$ of the root node of the resulting spanning BFS-tree, the identifier $v.\mathsf{Parent}$ of its father in this tree and an integer $v.\mathsf{Distance}$ which is the tree-distance between v and the root. This phase ends if the following condition holds in each node v:

Condition $st(v)$:

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

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 node, 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 square-face are included in different sets E_1 and E_2. For a vertex v, we assume that the edges containing v as an end-vertex are numbered $0, \ldots, deg(v) - 1$ in the order in which they appear in the counterclockwise traversal, so that two edges incident to v and belonging to a common square have numbers of different parity. For each $v \in V$, the variable $v.\mathsf{Color}$ equals i if all edges incident to v which appear at even positions in the adjacency list of v belong to E_i and the remaining edges belong to E_{3-i}. Each vertex v runs the following algorithm:

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

We say that the condition $col(v)$ *is satisfied by a vertex* $v \in V$ if none of the preconditions of the three previous actions is satisfied. Then Phase 2 terminates if the condition $col(v)$ is satisfied by all vertices $v \in V$ of G. At this time each vertex v knows the list $v.\mathsf{Edge-list}_i$ of its neighbors in G_i.

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 $v.\mathsf{Parent}_i$ which will denote the father of the vertex v in the tree ST_i. The algorithm consists in "correcting" the spanning tree of Phase 1 by replacing $v.\mathsf{Parent}$ with a vertex belonging to $v.\mathsf{Edge\text{-}list}_i$ if this list contains a vertex located at the same distance to the root as the vertex $v.\mathsf{Parent}$. 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:

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

We say that the condition $st_i(v)$ *is satisfied by the vertex* v if none of the preconditions of the previous actions is satisfied. 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 node $v \in V$ knows its father $v.\mathsf{Parent}_i$ 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 MEDIANTREE 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 $v.\mathsf{b_i}$ 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 $Med_\pi(G)$ of G is formed by all vertices v of G for which $v.\mathsf{b_1} \wedge v.\mathsf{b_2}$ is true.

Theorem 3.2 *The algorithm* MEDIANEVENSQUAREGRAPH *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 the round $i+1$, the variable $w.\mathsf{Color}$ of its father w, which is at distance $i-1$ from the root, has already been correctly computed (by induction hypothesis). Thus the rules of Phase 2 correctly compute the value of $v.\mathsf{Color}$ using that of $w.\mathsf{Color}$. Since two edges incident to the same vertex v and belonging to the same square have numbers of different parity in the degree 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 $u' \neq v$. It was shown in [22] for graphs more general than squaregraphs that, if u and v are adjacent in G, then their fathers u' and v' are also adjacent. Since $d_G(v',r) = k-1$, by 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 square 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 in Phase 3 the trees ST_1 and ST_2, it suffices that each vertex is activated at this phase once. As to the Phase 4, the algorithm MEDIANTREE stabilizes in $O(e)$ rounds, where e is the largest eccentricity of a median vertex of G [18], 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. Most importantly, Proposition 2.5 establishes that MEDIANEVENSQUAREGRAPH correctly computes $\mathrm{Med}(G)$. □

Remark. Note that, given a spanning tree of an even squaregraph and unique identifiers, it would be possible to reconstruct locally a map representing the topology of the graph and then compute *internally* the median set. But it would require much more than $O(\log n)$ bits per node and therefore it is not a satisfactory solution.

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 applies the majority rule for computing medians and (ii) our graphs G isometrically embed into the Cartesian product

of two trees. 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 [33] 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, 18]), i.e., without the isometric embedding into the product of trees?

Question 2: Our algorithm for computing the medians of even squaregraphs requires that every node has a unique identifier. Is it possible to design an algorithm not requiring this property, i.e., assuming that all nodes are anonymous?

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 and which uses $O(k)$ variables per node?

Question 4: Rephrasing the first open problem of Gerstel and Zaks [33], 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. [18] 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 [23]).

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, 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 Processing 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, pp. 652–661, 1993.

[9] Bandelt H.-J., Barthélemy J.-P., Medians in median graphs, Discr. Appl. Math. **8** (1984) 131-142.

[10] Bandelt H.-J., Chepoi V., Graphs with connected medians, SIAM J. Discrete Math. **15** (2002) 268-282.

[11] Bandelt H.-J., Chepoi V., Metric graph theory and geometry: a survey, Contemporary Mathematics (to appear).

[12] Bandelt H.-J, Chepoi V., Eppstein D., Ramified rectilinear polygons (in preparation).

[13] Barthélemy J.P., Janowitz M.F., A formal theory of consensus, SIAM J. Discr. Math. **4** (1991) 305-322.

[14] 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.

[15] Barthélemy J.-P., Monjardet B., The median procedure in cluster analysis and social choice theory, Math. Soc. Sci. **1** (1981) 235-268.

[16] Buckley F., Harary F., *Distances in Graphs,* Redwood City, CA: Addison-Wesley, 1990.

[17] Blair J.R.S, Manne F., Efficient self-stabilizing algorithms for tree networks, Tech. Rept. 232, Univ. Bergen, 2002.

[18] Bruell S.B., Ghosh S., Karaata M.H., Pemmaraju S.V., Self-stabilizing algorithms for finding centers and medians of trees, SIAM J. Computing **29** (1999) 600–614.

[19] Boldi P., Vigna S., An effective characterization of computability in anonymous networks, In DISC 2001, Springer LNCS, vol. 2180, pp. 33–47, 2001.

[20] Chen N.S., Yu H.P., Huang S.T., A self-stabilizing algorithm for constructing spanning trees, Inf. Proc. Let. **39** (1991) 147–151.

[21] Chalopin J., *Algorithmique Distribuée, Calculs Locaux et Homomorphismes de Graphes,* PhD Thesis Université Bordeaux I (2006).

[22] Chepoi V., Graphs of some CAT(0) complexes, Adv. Appl. Math. **24** (2000) 125–179.

[23] Chepoi V., Dragan F., Vaxès Y., Center and diameter problem in planar quadrangulations and triangulations, In SODA'02, pp. 346-355, 2002.

[24] Chepoi V., Fanciullini C., Vaxès Y., Median problem in some plane triangulations and quadrangulations, Computational Geometry **27** (2004) 193–210.

[25] Datta A.K., Derby J.L., Lawrence J.E., Tixeuil S., Stabilizing hierarchical routing, J. Interconnexion Networks **1** (2000) 283–302.

[26] Delat S., Nguyen D., Tixeuil S., Stabilité et auto-stabilisation du routage inter-domaine dans internet, In *RIVF*, pp. 139–144, 2003.

[27] Dijkstra E. W., Self-stabilizing systems in spite of distributed control, Communications of ACM **17** (1974) 643–644.

[28] Dolev S., *Self-stabilization,* MIT Press, 2000.

[29] Dolev S., Israeli A., Moran S., Self-stabilization of dynamic systems assuming only read/write atomicity, In PODC'90, pp. 103–118, 1990.

[30] Dress A., Scharlau R., Gated sets in metric spaces, Aequat. Math. **34** (1987) 112–120.

[31] Flocchini P., Mans B., Santoro N., Sense of direction: formal definitions and properties, In SIROCCO'95, pp. 9–34, 1995.

[32] Gärtner F., A survey of self-stabilizing spanning-tree construction algorithms (2003).

[33] Gerstel O., Zaks S., A new characterization of tree medians with applications to distributed algorithms, Networks **24** (1994) 23-29.

[34] Ghosh S., Gupta A., Pemmaraju S.V., A self-stabilizing algorithm for the maximum flow problem, Distributed Computing **10** (1997) 167-180.

[35] Ghosh S., Karaata M.H., A self-stabilizing algorithm for coloring planar graphs, Distributed Computing **7** (1993) 55–59.

[36] Goldman A.J., Witzgall C.J., A localization theorem for optimal facility placement, Transp. Sci. **4** (1970) 406–409.

[37] Gouda M.G., Herman T., Stabilizing unison, Inf. Proc. Let. **35** (1990) 171–175.

[38] Gouda M.G., Schneider M., Maximum flow routing, In Proc. of the Second Workshop on Self-Stabilizing Systems, pp. 1–13, 1995.

[39] Hedetniemi S.T., Jacobs D.P., Srimani P.K., Maximal matching stabilizes in time o(m), Inf. Proc. Let. **80** (2001) 221–223.

[40] Hendry G.R.T, On graphs with prescribed medians, I, J. Graph Th. **9** (1985) 477–481.

[41] Herman T., Ghosh S., Stabilizing phase-clocks, Inf. Proc. Let. **54** (1995) 259–265.

[42] 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.

[43] Kariv O., Hakimi S.L., An algorithmic approach to network location problems, I,II, SIAM J. Appl. Math. **37** (1979) 513–538, 539–560.

[44] Korach E., Rotem D., Santoro N., Distributed algorithms for finding centers and medians in networks, ACM Trans. Program. Lang. Syst. **6** (1984) 380–401.

[45] Mulder H.M., *The Interval Function of a Graph,* Math. Centre Tracts 132, Amsterdam, 1980.

[46] Nesterenko M., Mizuno M., A quorum-based self-stabilizing distributed mutual exclusion algorithm, J. Parallel Distrib. Comput. **62** (2002) 284–305.

[47] Santoro N., Sense of direction, topological awareness, and communication complexity, ACM SIGACT News **16** (1984) 50–56.

[48] Shukla S., Rosenkrantz D., Ravi S., Developing self-stabilizing coloring algorithms via systematic randomization, In Proc. of the Int. Workshop on Parallel Proc., pp 668–673, 1994.

[49] 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.

[50] Tansel B.C, Francis R.L, Lowe T.J., Location on networks: a survey, Management Sci. **29** (1983) 482–511.

[51] van de Vel M.L.J., *Theory of Convex Structures,* North-Holland, Amsterdam, 1993.