Median problem in some plane triangulations and quadrangulations

Victor Chepoi, Clémentine Fanciullini, and Yann Vaxès

Laboratoire d'Informatique Fondamentale,
Université de la Méditerranée,
Faculté des Sciences de Luminy,
F-13288 Marseille Cedex 9, France

Abstract. In this note, we present linear time algorithms for computing the median set of plane triangulations with inner vertices of degree ≥ 6 and plane quadrangulations with inner vertices of degree ≥ 4.

1. Introduction

Given a finite, connected graph $G=(V,E)$ endowed with a non-negative weight function $\pi(v)(v \in V)$ the *median set* $\mathrm{Med}(\pi)$ consists of all vertices x minimizing the total weighted distance

$$F_\pi(x)=\sum_{v\in V}\pi(v)d(v,x).$$

Finding the median set of a graph, or, more generally, of a network or a finite metric space is a classical optimization and algorithmic problem with many practical applications. The weighted version of the median problem is one of the basic models in facility location (where it is sometimes called the Fermat-Weber problem); see for example [24]. It arises with majority consensus in classification and data analysis [5, 9, 23], where the median points are usually called Kemeny medians. Algorithms for locating medians in graphs are especially useful in the areas of transportation and communication in distributed networks: placing a common resource at a median minimizes the cost of sharing the resource with other locations or the total time of broadcasting messages. Recently, motivated by a heuristic for reconstructing discrete sets from projections presented in [8], the medians sets of polyominoes and some other special subsets of the square grid have been investigated in [19, 22] (in [17], similar questions have been considered in the more general setting of linear metrics). Finally, [12] provides efficient algorithms for the approximate computation of the values of the function $F_\pi(x)$ for points in $\mathbb{R}^n$ endowed with the Euclidean distance.

There are known several algorithms [24] for finding medians of graphs, but only for trees the classical majority rule can be implemented in linear time [20, 21, 27]: given a tree T and an edge $e = xy$, the median set is contained in the heaviest of two subtrees T_x, T_y defined by this edge (if the subtrees have the same weight, then both x and y are medians). This is so because $F_\pi(x) - F_\pi(y)$ equals the difference between the weights of T_y and T_x and the fact that every local minimum of the function F_π is a global minimum; for more details, see [24]. Later, using techniques from computational geometry this approach has been extended to design an efficient algorithm for the median problem on simple rectilinear polygons with an intrinsic l_1-metric. More recently, similar ideas were used in [2, 14] to develop simple (but very nice) self-stabilizing algorithms for finding medians of trees; see also [1, 4] for efficient algorithms for maintaining medians in dynamic trees. Last but not least, [7] characterizes the graphs in which all local medians are global medians for each weight function π (by a *local median* one means a vertex x such that $F_\pi(x)$ does not exceed $F_\pi(y)$ for any neighbor y of x). In particular, it is shown in [7] that these graphs can be recognized in polynomial time and that they are exactly the graphs in which all median sets induce connected or isometric subgraphs.

In this note, we describe linear time algorithms for computing the median sets in two classes of face regular plane graphs. Namely, we consider plane triangulations with inner vertices of degree at least six (called *trigraphs*) and plane quadrangulations with inner vertices of degree at least four (called *squaregraphs*); see Fig.1 for examples. Particular cases of these graphs are, the subgraphs of the regular triangular and square grids which are induced by the vertices lying on a simple circuit and inside the region bounded by this circuit (the latter comprises the graphs from [19, 22]). Notice that these classes of plane graphs are particular instances of bridged and median graphs, two classes of graphs playing an important role in metric graph theory. The trigraphs has been introduced and investigated in [6] where they are the basic building stones in the construction of weakly median graphs. The present paper continues the line of research of [16] where linear time algorithms for computing the diameter and the center of these graphs have been proposed (the terms "trigraph" and "squaregraph" are owed from that paper). It should be noted in passing that, due to the different nature of the objective functions (minsum and minmax), the method used here is completely different from that of [16]. Nevertheless, both papers have the same flavor of developing a kind of computational geometry in plane graphs based on natural convexity and metric properties of graphs in question.

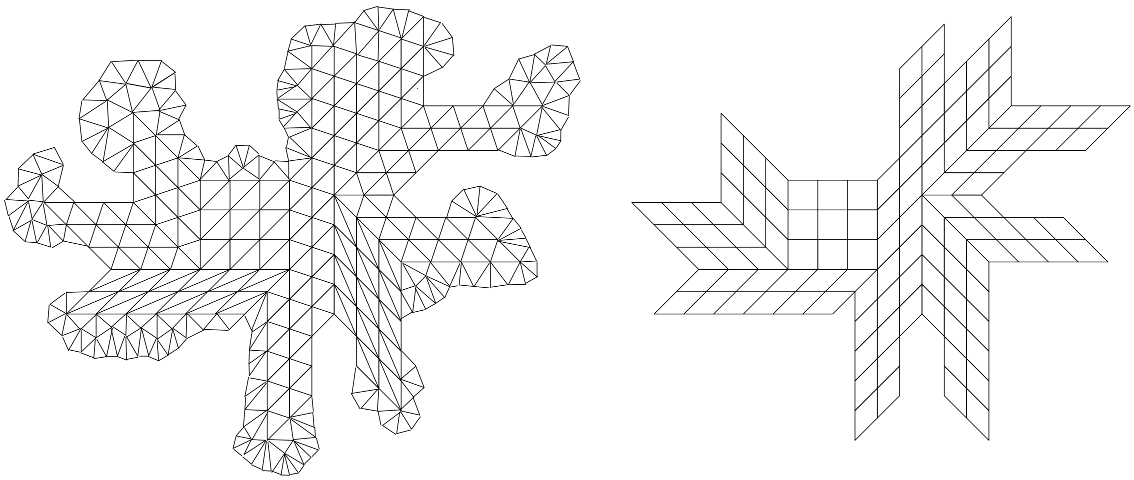


Fig. 1

By replacing every inner face of a plane triangulation by an equilateral triangle of side 1, one obtain a two-dimensional (pseudo-)manifold embedded in some high dimensional space. Analogously, one can define such a manifold if one replace every inner face of a plane quadrangulation by a unit square. (For squaregraphs and trigraphs such manifolds can be effectively constructed via an isometric embedding of these graphs into hypercubes and half-cubes as is done in [6].) The resulting manifolds can be endowed in a natural way with an intrinsic Euclidean metric as is explained in [13]. Now, the trigraphs and the squaregraphs are precisely the plane triangulations and quadrangulations for which these surfaces have intrinsic metric of non-positive curvature [10, 13], therefore they may arise, among others, in the following type of applications. Recently, [25, 11] proposed a new technique (called Isomap) of data analysis (as an alternative to principal component analysis or multidimensional scaling) which, on the base of easily measured local metric information, aims to build the underlying global geometry of data sets in the form of a low-dimensional structure embedded in high-dimensional data space. Isomap deals with data sets of $\mathbb{R}^n$ which are assumed to lie on a smooth manifold M of low dimension. The crucial stage of the method consists in approximating the unknown geodesic distance in M between data points in terms of the graph distance with respect to some graph G constructed on the data points. Hence, when M is 2-dimensional and has non-positive curvature it is likely that the resulting graph G will be a trigraph or a squaregraph. On the other hand, the medians computed with respect to the distance function of G can be viewed as a natural extension of the usual notion of median used in data analysis and statistics. Finally, notice that the terrains can be viewed as particular instances of such pseudo-manifolds, and that efficient algorithms for the center problem on terrains have been proposed in [3].

Our method of computing $\mathrm{Med}(\pi)$ for trigraphs is based on the following. From the results of [7] follows that in trigraphs the function F_π is unimodal for all the choices of weights. As in case of trees, solely this important fact does not yield a linear time algorithm because computing $F_\pi(x)$ for a single vertex x already needs linear time. Instead, for trigraphs we show how to compute in total linear time the differences $\Delta(x,y) := F_\pi(x) - F_\pi(y)$ for all edges xy of G. Using this information, we define the directed graph $\overrightarrow{G}_\pi$ in which the edge xy of G is replaced by the arc $\overrightarrow{yx}$ if $\Delta(x,y) < 0$ and by the arc $\overrightarrow{xy}$ if $\Delta(x,y) > 0$; no arc between x and y is defined if $\Delta(x,y) = 0$. Due to the unimodality of the function F_π, the median set $\mathrm{Med}(\pi)$ consists of all vertices having no outgoing arcs in $\overrightarrow{G}_\pi$. Clearly, these vertices can be found in linear time by traversing $\overrightarrow{G}_\pi$. Notice also that with these differences at hand we can easily compute in total linear time all values of the function F_π in the following way: compute $F_\pi(c)$ for some vertex c and construct a Breadth-First-Search tree rooted at c. For each vertex v let v' be its father in this tree. Now, if $F_\pi(v')$ has been already computed, then set $F_\pi(v) := F_\pi(v') + \Delta(v,v')$, and continue the traversal of the tree.

For squaregraphs, the majority rule together with a simple but nice trick yield a divide-and-conquer linear time algorithm for computing a part of the median set $\mathrm{Med}(\pi)$.

The paper is organized as follows. In the next section, we recall some necessary notions and formulate some auxiliary results. As a warm-up, in Section 3 we present the algorithm for computing medians of squaregraphs. In Section 4 we describe the main contribution of this note – a linear time algorithm for the median problem in trigraphs.

2. Preliminaries

All graphs $G = (V, E)$ occurring in this note are connected, finite, and undirected. Since computing the median set of a graph can be reduced in linear time to computing the median sets inside its 2-connected components [21], we may assume without loss of generality that G itself is 2-connected. In a graph G, the *length* of a path from a vertex v to a vertex u is the number of edges in the path. The *distance* $d(u, v)$ from u to v is the length of a minimum length (u, v)-path and the *interval* $I(u, v)$ between these vertices is the set $I(u, v) = \{w \in V : d(u, v) = d(u, w) + d(w, v)\}$. A subset $S \subseteq V$ is called *convex* if $I(u, v) \subseteq S$ whenever $u, v \in S$, and *gated* [21] if for each $v \notin S$ there exists a (necessarily unique) vertex $v' \in S$ (the *gate* of v in S) such that $v' \in I(v, u)$ for every $u \in S$. For a weight function π and a subset S of vertices, let $\pi(S) = \sum_{s \in S} \pi(s)$ denote the weight of S. In particular, $\pi(V)$ denotes the total weight of vertices of G. Obviously, we can suppose that $\pi(V)$ is known in advance (otherwise, it can be easily computed in linear time).

For an edge uv of a graph G, let

$$W(u, v) = \{x \in V : d(u, x) < d(v, x)\},$$

$$W(v, u) = \{x \in V : d(v, x) < d(u, x)\}.$$

The following well-known lemma is trivial but crucial:

Lemma 1. *For every weight function π and every edge uv of G we have*

$$F_\pi(u) - F_\pi(v) = \pi(W(v, u)) - \pi(W(u, v)).$$

Indeed, a vertex x of $W(v, u)$ contributes with $+\pi(x)$ to $F_\pi(u) - F_\pi(v)$, a vertex x of $W(u, v)$ contributes with $-\pi(x)$ to this difference, while every vertex equidistant to u and v does not contribute at all. Summing up over all vertices of G, we obtain the right-hand side.

In view of Lemma 1, in order to construct the oriented graph $\overrightarrow{G}_\pi$ efficiently, we must be able to compute $\pi(W(u, v))$ and $\pi(W(v, u))$ for all edges uv of G. If G is a bipartite graph, then $W(u, v) \cup W(v, u) = V$, therefore it is enough to find the weight of only one of these complementary sets. Moreover, if G is a squaregraph, then $W(u, v)$ and $W(v, u)$ are gated sets, because the squaregraphs are median graphs; cf. [26]. From the results of [21] follows that in this case $\mathrm{Med}(\pi) \subseteq W(u,v)$ if and only if $\pi(W(u,v)) > \pi(W(v,u))$. In case of trigraphs, the sets $W(u,v)$ and $W(v,u)$ are convex, but they no longer cover the whole vertex-set of G. Nevertheless, these sets extend to two couples of complementary convex sets which we will call *half-planes*. More precisely, in [6] it is shown that any two adjacent vertices u and v of a trigraph G are separated by exactly two distinct pairs of complementary half-planes H'_u, H'_v and H''_u, H''_v such that $u \in W(u,v) = H'_u \cap H''_u$ and $v \in W(v,u) = H'_v \cap H''_v$. In Section 4 we will show that the half-planes of G separating u and v have a geometric nature which allow to proceed them efficiently.

To conclude this section, notice that in subsequent algorithms every trigraph or squaregraph G is represented by a *doubly-connected edge list*; for precise definition and details see [18]. We recall here only a few things about this data structure. Since every edge of G bounds two faces, it is convenient to view the different sides of an edge as two distinct *half-edges*. The two half-edges $\overrightarrow{xy}$ and $\overrightarrow{yx}$ we get for an edge xy are called *twins* (so that $twin(\overrightarrow{xy}) = \overrightarrow{yx}$ and $twin(\overrightarrow{yx}) = \overrightarrow{xy}$). The half-edges bounding the outer face ∂G are oriented so that ∂G is traversed in clockwise order. On the other hand, the half-edges of every inner face are oriented so that the face is traversed in counterclockwise order. The half-edge record of a half-edge $\overrightarrow{e}$ stores a pointer to its origin, a pointer to its twin, a pointer to the incident face, and two pointers $next(\overrightarrow{e})$ and $prev(\overrightarrow{e})$ to the next and the previous edges on the boundary of incident face.

3. Computing median sets in squaregraphs

Let G be a squaregraph and let uv be an edge on the outer face of G. As we noticed above, the half-planes $W(u,v)$ and $W(v,u)$ separating u and v are gated sets. Moreover, the subgraph P induced by all vertices of $W(u,v)$ having a neighbor in the set $W(v,u)$ is a gated path (analogously one can define the gated path $Q \subseteq W(v,u)$). Notice that there is a natural isomorphism between the paths P and Q. For every vertex $x \in P$, let F_x consists of all vertices of $W(v,u)$ whose gate in $W(u,v)$ is the vertex x and call this set the *fiber* of x. Analogously define the fiber F_y of every vertex $y \in Q$. The subgraph induced by $P \cup Q$ is a strip consisting of one or several inner faces of G. This strip and the paths P and Q can be easily constructed in $O(|P| + |Q|)$ time starting from the edge uv and the unique inner face containing this edge.

Now, the median set Med(π) is contained in $W(u,v)$ if $\pi(W(u,v)) > \frac{1}{2}\pi(V)$ and is contained in $W(v,u)$ if $\pi(W(v,u)) > \frac{1}{2}(\pi(V))$ [21]. Finally, if $\pi(W(u,v)) = \pi(W(v,u))$, then Med($\pi$) intersects both sets $W(u,v)$ and $W(v,u)$. Since Med(π) is convex and therefore gated, it will have common vertices with both P and Q. Therefore we can continue the search in the subgraph induced by $W(u,v)$ in the first case, in the subgraph induced by $W(v,u)$ in the second case, and in the strip $P \cup Q$ in the third case. The respective subgraph G' is endowed with a new weight function π' defined in the following way. In the first case, define π' on $W(u,v)$ by setting $\pi'(x) := \pi(x)$ for every $x \in W(u,v) - P$ and $\pi'(x) := \pi(F_x)$ for every $x \in P$. Analogously, in the second case define π' on $W(v,u)$ by setting $\pi'(y) := \pi(y)$ for every $y \in W(v,u) - Q$ and $\pi'(y) := \pi(F_y)$ for every $y \in Q$. Finally, in the third case define π' on $P \cup Q$ by setting $\pi'(x) := \pi(F_y)$ and $\pi'(y) = \pi(F_x)$, where $x \in P$ and $y \in Q$ are adjacent to each other. Then one can see that Med(π')=Med(π) in first and second cases and that Med(π') $\subseteq$ Med(π) in third case. In the latter case Med(π') can be easily computed applying the majority rule to the resulting strip.

In order to implement this algorithm in linear time, at each step we have to decide in which of three cases we are by traversing a part of the current graph G proportional in size to the part which will be removed from further consideration. For example, if Med(π) is contained in $W(u,v)$, then we have to decide this in time $O(|W(v,u)|)$. This is possible using the following trick: perform in parallel the Breadth-First-Search on the sets $W(u,v)$ and $W(v,u)$ starting from the paths P and Q, respectively, and stop when one of the sets will be completely traversed. (This can be easily done by alternatively searching each of the half-planes according to BFS). During these BFS traversals, we add the weight of the current vertex v to the weight of the half-plane and the fiber F_x containing it. For this notice that v will be in the same half-plane and the same fiber as its father in the respective BFS tree. Suppose without loss of generality that the search of $W(v,u)$ was completed first. If $\pi(W(v,u)) < \frac{1}{2}\pi(V)$, then Med($\pi$) is contained in $W(u,v)$ and we spent $O(|W(v,u)|)$ time to decide this and to construct the weight function π'. Since finding the median set in $W(u,v)$ will take $O(|W(u,v)|)$ time, we conclude that the overall time is $O(|V|)$. On the other hand, if $\pi(W(v,u)) \geq \frac{1}{2}\pi(V)$, then Med($\pi$) is contained in $W(v,u)$. In this case, we continue the traversal of $W(u,v)$ in order to compute the weights of all fibers of this set. Since $|W(u,v)| \geq |W(v,u)|$, we spent $O(|W(u,v)|)$ time to conclude that the search of median vertices should be continued in $W(v,u)$ or in $P \cup Q$. All this shows that employing this simple approach we can find at least one part of $\mathrm{Med}(\pi)$ is linear time. To compute the whole median set either we have to expand the computed part in a careful way by taking into account that $\mathrm{Med}(\pi)$ is an interval or to adopt an approach similar to that for trigraphs presented in the next section.

4. Computing median sets in trigraphs

Throughout this section, $G = (V, E)$ is a trigraph stored in the form of a doubly-connected edge list whose outer face is traversed clockwise. Notice that the outer face of each other type of regions occurring below (half-planes, sectors, cones, and stripes) is also traversed clockwise. Trigraphs, as a subclass of bridged graphs, have the following characteristic property: balls and, more generally, neighborhoods of convex sets in such graphs are convex. (Recall that the *ball* $B_r(c)$ of radius r and center c consists of all vertices at distance at most r from c. The *neighborhood* $N(S)$ of a set S consists of S and all vertices of $V - S$ having a neighbor in S.) From this property one can easily conclude that trigraphs do not have induced 4- and 5-cycles. On the other hand, as is shown in [6], trigraphs do not have 4-cliques K_4 and the graph $K_{1,1,3}$ consisting of three triangles having an edge in common. Another useful property of these graphs established in [6] is the following

Triangle condition: for any three vertices u, v, w of G with $1 = d(v,w) < d(u,v) = d(u,w)$ there exists a common neighbor x of v and w such that $d(u,x) = d(u,v) - 1$.

For further results and references on trigraphs and bridged graphs see [6].

4.1. Half-planes, sectors, cones, zips. Let $(H_1, H'_1), \ldots, (H_m, H'_m)$ be the pairs of complementary half-planes of G. Denote by P_i and P'_i the subgraphs induced by the vertices of H_i and H'_i which have neighbors in the complementary half-plane (i.e., in H'_i and H_i, respectively).

Lemma 2. *P_i and P'_i are convex paths of G.*

Proof. P_i is the intersection of two convex sets H_i and $N(H'_i)$, therefore it is convex (analogously one deduce that P'_i is convex). Since G is K_4-free and P'_i is convex, every vertex of P_i has one or two adjacent neighbors in P'_i.

Pick two adjacent vertices x, y of P_i, and let x' and y' be their mutually closest neighbors in P_i'. We assert that $x' = y'$. Suppose not. If x' and y' are adjacent, then we obtain a 4-cycle (x, y, y', x') which cannot be induced. But if x is adjacent to y' or if y is adjacent to x' we obtain a contradiction with the choice of x' and y'. Now, if x' and y' are not adjacent, then $d(x', y') = 2$, because otherwise $x, y \in I(x', y')$, contrary to the convexity of P_i'. Pick a common neighbor z' of x' and y', which necessarily belongs to $I(x', y') \subseteq P_i'$. The 5-cycle (x, y, y', z', x') cannot be induced, whence z' is adjacent to x and y, again yielding a contradiction with the choice of x' and y'. Hence each pair of adjacent vertices of P_i has a common neighbor in P_i'.

Since P_i is convex, it is also a trigraph. Therefore to show that P_i is a path it suffices to show that it does not contain 3-cycles and vertices of degree 3. Suppose by way of contradiction, that P_i contains three pairwise adjacent vertices x, y, z. If they have a common neighbor in P_i' we will get a K_4, a contradiction. So let $z' \in P_i'$ be the common neighbor of x and y, $y' \in P_i'$ be the common neighbor of x and z, and $x' \in P_i'$ be the common neighbor of y and z. The vertices x', y', and z' are pairwise adjacent because P_i' is convex. Now, in order to avoid an induced 4-cycle generated by x, y, x', y', either x and x' are adjacent or y and y' are adjacent. In both cases, we obtain a 4-clique, which is impossible. Thus P_i and P_i' induce acyclic subgraphs. Finally, assume by way of contradiction that P_i contains a $K_{1,3}$, i.e., a vertex x adjacent to three other vertices y, z, v. Now, if we consider the common neighbors y', z', v' in P_i' of y and x, z and x, and v and x, respectively, the convexity of P_i and P_i' implies that these vertices must be distinct and pairwise adjacent. This contradicts the fact that P_i' does not contains 3-cycles. This final contradiction shows that indeed P_i and P_i' are convex paths. $\square$

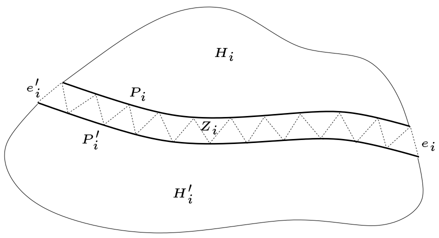
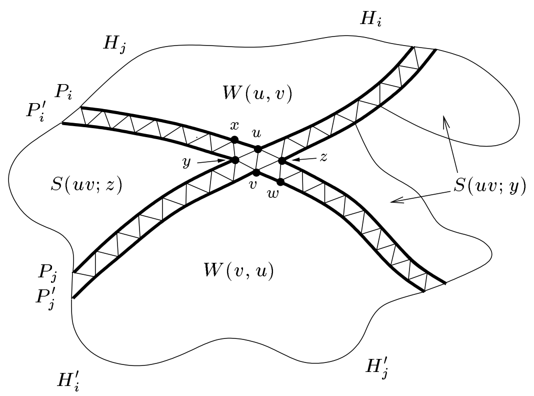


Fig. 2

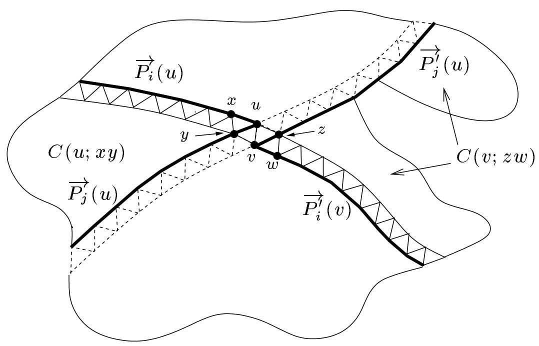
We call the convex paths P_i and P_i' the *lines* of the half-planes H_i and H_i'. Denote by Z_i the partial subgraph of G comprising all edges with one end in P_i and another one in P_i'

and, due to its form, call Z_i a *zip*; see Fig. 2. for an illustration. A *strip* S_i is the union of all inner faces of G sharing two edges with the zip Z_i. Notice that every zip Z_i shares two edges e_i and e'_i with ∂G, so that S_i lies to the left of the half-edge of e_i which bounds ∂G. Below we will show that the zip Z_i can be reconstructed in a canonical way starting from the half-edge $\overrightarrow{e_i}$.

As we noticed above, every two adjacent vertices u and v of G are separated by exactly two distinct pairs of complementary half-planes H_i, H'_i and H_j, H'_j, where $u \in H_i \cap H_j$ and $v \in H'_i \cap H'_j$. Then, as we noticed above, $H_i \cap H_j = W(u,v)$ and $H'_i \cap H'_j = W(v,u)$. Two other intersections $H_i \cap H'_j$ and $H'_i \cap H_j$ are called *sectors* and denoted by $S(uv;y)$ and $S(uv;z)$, respectively, where y and z are the common neighbors of u and v (if the edge uv belongs to the outer face ∂G, then only one of two sectors is defined). One can easily see that $W(u,v) = H_i - S(uv;z)$ and $W(v,u) = H'_j - S(uv;z)$, i.e., in order to find the weight of the sets $W(u,v), W(v,u) (uv \in E)$ it suffices to compute the weights of all half-planes and sectors; see Fig.3a. To do this, we find more appropriate to perform all computations with objects slightly different from sectors, which we call cones and define below.



(a)



(b)

Fig. 3

Denote by $\overrightarrow{P_i}(u)$ and $\overleftarrow{P_i}(u)$ the sub-paths of the path P_i (with respect to the clockwise traversal of ∂H_i) such that $\overrightarrow{P_i}(u) \cap \overleftarrow{P_i}(u) = \{u\}$ and $\overrightarrow{P_i}(u) \cup \overleftarrow{P_i}(u) = P_i$. Call the oriented paths $\overrightarrow{P_i}(u)$ and $\overleftarrow{P_i}(u)$ *u-rays.* (Analogously one can define the u-rays $\overrightarrow{P_j}(u), \overleftarrow{P_j}(u)$ and the v-rays $\overrightarrow{P'_i}(v), \overleftarrow{P'_i}(v), \overrightarrow{P'_j}(v)$, and $\overleftarrow{P'_j}(v)$.) Notice that every inner half-edge having origin u extends to a unique u-ray. Let x be the neighbor of u in the ray $\overrightarrow{P_i}(u)$. Set $C(u; xy) := S(uv; y) \cup \overrightarrow{P_i}(u)$ and call the set $C(u; xy)$ a *cone* with *apex* u and *generator* xy, see Fig.3b for an illustration. The u-rays $\overrightarrow{P_i}(u)$ and $\overrightarrow{P_j}(u)$ are the bounding rays of $C(u; xy)$. Analogously, if w is the neighbor of v in the ray $\overrightarrow{P'_i}(v)$, we define the cone $C(v; zw) := S(uv; z) \cup \overrightarrow{P'_i}(v)$ with apex v, generator zw and bounding rays $\overrightarrow{P'_i}(v)$ and $\overrightarrow{P'_j}(v)$. In order to treat degenerated cases, it will be convenient to extend the notion of a cone to the case when $u \in \partial G$ and $x = y \in \partial G$; we denote such a cone by $C(u; xx)$ or $C(u; yy)$ and call it *degenerated.* Notice that a degenerated cone $C(u; xx)$ may be viewed as an usual cone $C(u; xy)$ in the trigraph obtained from G by adding a new vertex y and making it adjacent to two consecutive vertices u, x of ∂G (analogously, $C(u; yy)$ may be viewed as the cone $C(u; xy)$ in the trigraph obtained from G by adding a new vertex x adjacent to u and y). Hence, in the sequel it suffices to show how to deal with non-degenerated cones only.

We will establish below that every half-plane H_i can be represented as a union of cones having their apices at the origin of $\overrightarrow{e_i}$, therefore $\pi(H_i)$ (and therefore $\pi(H'_i)$)) can be computed provided we know the weights of the cones of G.

4.2. The algorithm. Summarizing our discussion, we outline the following algorithm for computing the median set $\mathrm{Med}(\pi)$ of a trigraph G, whose steps will be further detailed in subsections 4.3.-4.5.:

Algorithm MEDIAN SET

Input: A trigraph G in the form of a doubly-connected edge list and a weight function π

Output: The median set $\mathrm{Med}(\pi)$

1. Compute the zips Z_i, their strips S_i, and the lines P_i, P'_i $(i = 1, \ldots, m)$;
2. For $i = 1, \ldots, m$ and all vertices $u \in P_i, v \in P'_i$ compute the weights $\pi(\overrightarrow{P_i}(u))$, $\pi(\overleftarrow{P_i}(u))$, $\pi(\overrightarrow{P'_i}(v))$, $\pi(\overleftarrow{P'_i}(v))$ of the u- and v-rays;
3. Compute the weights $\pi(C(u; xy))$ of the cones $C(u; xy)$ of G, and then compute the weights $\pi(S(uv; x))$ of the sectors $S(uv; x)$ of G;

4. Compute the weights $\pi(H_i)$ and $\pi(H_i')$ of the half-planes H_i and H_i' $(i = 1, \ldots, m)$;

5. For each edge uv of G compute $\pi(W(u,v))$ and $\pi(W(v,u))$ as the difference between weights of a half-plane and a sector computed in steps 3 and 4;

6. Construct the graph $\overrightarrow{G_\pi}$;

7. Return the set $\mathrm{Med}(\pi)$ consisting of all vertices of G having no outgoing edges in $\overrightarrow{G_\pi}$.

4.3. Computing zips, strips, lines, and weights of rays. To perform this computation, we traverse the half-edges of ∂G in clockwise order. Let $\overrightarrow{e_i} = \overrightarrow{uv}$ be the current half-edge of ∂G, for which we aim to construct the zip Z_i and the lines P_i, P_i' (the strip S_i can be easily recovered from Z_i). More precisely, our algorithm will return one half-edge per edge of respective line, so that $\overrightarrow{P_i}$ will be an oriented path starting at u, $\overrightarrow{P_i'}$ will be an oriented path ending at v, while $\overrightarrow{Z_i}$ will keep the half-edges having the origin in P_i and the destination in P_i'.

Algorithm ZIP

Input: $\overrightarrow{e_i} \in \partial G$

Output: $\overrightarrow{Z_i}, \overrightarrow{P_i}, \overrightarrow{P_i'}$

$k := 0, \overrightarrow{Z_i} := \{\overrightarrow{e_i}\}, \overrightarrow{P_i} := \emptyset,$

$\overrightarrow{P_i'} := \emptyset, \overrightarrow{e} := twin(\overrightarrow{e_i})$

while $\overrightarrow{e} \notin \partial G$ do

  if k is even

   then

    add $next(\overrightarrow{e})$ to $\overrightarrow{P_i}$ and $prev(\overrightarrow{e})$ to $\overrightarrow{Z_i}$

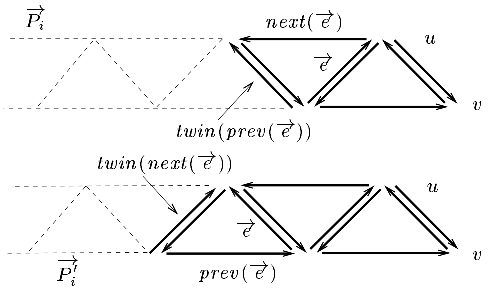
    set $k := k+1$ and $\overrightarrow{e} := twin(prev(\overrightarrow{e}))$

   else

    add $next(\overrightarrow{e})$ to $\overrightarrow{Z_i}$ and $prev(\overrightarrow{e})$ to $\overrightarrow{P_i'}$

    set $k := k+1$ and $\overrightarrow{e} := twin(next(\overrightarrow{e}))$

end do

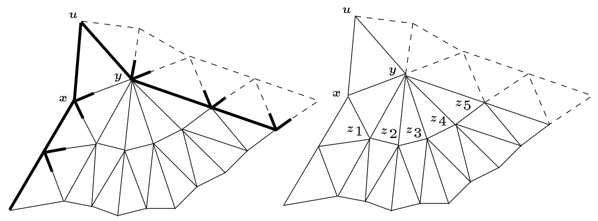


To establish the correctness of this algorithm, it suffices to show that P_i and P_i' induce convex paths of G. Indeed, this would imply that, removing the edges of Z_i, the connected components of the resulting graph are convex sets of G, therefore they are complementary half-planes. Consequently, we will deduce that Z_i is the zip of this pair of half-planes while P_i and P'_i are their lines. First notice that P_i and P'_i are paths because the algorithm alternatively adds half-edges to $\overrightarrow{P_i}$ and $\overrightarrow{P'_i}$. To show for example that the path P_i is convex, by Lemma 1 of [6] it is enough to prove that P_i is locally convex, i.e., if x, y, z are consecutive vertices of P_i, then x and z do not have other common neighbors in G. Suppose not, and let y' be such a common neighbor different from y. Let $\overrightarrow{yx'}$ and $\overrightarrow{zz'}$ be the half-edges which have been added to $\overrightarrow{Z_i}$ at the same iterations of the algorithm at which the half-edges $\overrightarrow{xy}$ and $\overrightarrow{yz}$ have been added to $\overrightarrow{P_i}$. Notice that $\overrightarrow{x'z'}$ is a half-edge of $\overrightarrow{P'_i}$. If x and z are adjacent, then these vertices together with x' and z' induce a 4-cycle, which is impossible. Hence x and z are not adjacent in G. Now, the vertices y and y' must be adjacent, otherwise the vertices x, y, z, y' induce a 4-cycle. If the half-edge $\overrightarrow{yy'}$ belongs to $\overrightarrow{Z_i}$, then we will obtain a contradiction with the algorithm, because the half-edges $\overrightarrow{xy}$ and $\overrightarrow{yz}$ will be added to $\overrightarrow{P_i}$ at two consecutive steps of the algorithm. Otherwise, the vertices x, y', z, z', x' will induce a 5-cycle, which is impossible. This shows that the paths P_i and P'_i are indeed locally convex, and therefore convex.

Finally notice that the complexity of this algorithm for a given boundary edge e_i is $O(|Z_i|+|P_i|+|P'_i|) = O(|Z_i|)$. Since every edge of G belongs to exactly two zips, summing up over the edges of ∂G, one conclude that the overall complexity of the algorithm is proportional to the number of edges of G, whence it is $O(|V|)$. Analogously, the overall size of the lists Z_i, P_i, and P'_i is also linear. Therefore, traversing the paths P_i and P'_i $(i = 1, \ldots, m)$ from the origin to the destination, in total linear time we will compute the weights $\pi(\overrightarrow{P_i}(u)), \pi(\overleftarrow{P_i}(u))$, $\pi(\overrightarrow{P'_i}(v)), \pi(\overleftarrow{P'_i}(v))$ for all vertices $u \in P_i$ and $v \in P'_i$.

4.4. Computing the weights of cones and sectors. Let $C(u; xy)$ be a cone of G bounded by the u-rays $\overrightarrow{P_i}(u)$ and $\overrightarrow{P_j}(u)$ (as we noticed above, one may assume that the cone $C(u; xy)$ is non-degenerated). First observe that the rays of $C(u; xy)$, for example $\overrightarrow{P_i}(u)$, can be constructed in the following way. Start by inserting the half-edge $\overrightarrow{ux}$ in $\overrightarrow{P_i}(u)$ and set $s := x$. At each step, given a current vertex s, turn counterclockwise around s starting from the half-edge next to the half-edge lastly inserted in $\overrightarrow{P_i}(u)$, then leave two edges incident to s and insert in $\overrightarrow{P_i}(u)$ the half-edge $\overrightarrow{ss'}$ of the third edge. Set $s := s'$, and repeat while s is an inner vertex of G. The path $\overrightarrow{P_j}(u)$ is constructed analogously. The single difference is that in the case of $\overrightarrow{P_i}(u)$ the two edges we leave at each iteration will belong to the cone $C(u; xy)$, while in case of $\overrightarrow{P_j}(u)$ they will be outside this cone; see Fig.4a.

Let z_0 be the neighbor of x in $\overrightarrow{P_i}(u)$ (if it exists). Then x may be adjacent to only one other vertex of $C(u;xy)$. On the other hand, the vertex y may have several neighbors in $C(u;xy)$ different from u and x, which we denote by $z_1, z_2, \ldots, z_p$. Notice that if z_0 exists, then x has another neighbor in $C(u;xy)$, namely z_1, and in this case z_1 and z_0 are adjacent. Since G is 2-connected, either the vertices $z_0, z_1, \ldots z_p$ induce a path of G or there exists an index $0 \leq k < p$ such that z_k and z_{k+1} are not adjacent and each of $z_0, \ldots, z_k$ and $z_{k+1}, \ldots, z_p$ induces a path of G; see Fig.4b. Special cases occur when z_0 does not exist or $k = p - 1$.



(a) (b)

Fig. 4

We continue with a formula expressing $\pi(C(u;xy))$ via the weights of the cone $C(x;z_0z_1)$ and of the cones of the form $C(y;z_jz_{j+1})$. First observe that all these cones belong to $C(u;xy)$: this follows from the way the bounding rays of a cone were constructed above. More precisely, we obtain

$$C(u;xy) = \{u\} \cup C(x;z_0z_1) \cup (\cup_{i=1}^{p-1} C(y;z_iz_{i+1})).$$

Some cones in this formula may overlap. However, two cones whose generators are not incident have only the apex y in common. On the other hand, the intersection of the cones $C(x;z_0z_1)$ and $C(y;z_1z_2)$ is again a cone. Finally, the intersection of two consecutive non-empty cones $C(y;z_{j-1}z_j)$ and $C(y;z_jz_{j+1})$ is a y-ray. (All this follows from the definition and the form of cones.) Hence, if z_0 exists and the vertices $z_0, z_1, \ldots, z_p$ induce a path, then we obtain the following inclusion-exclusion formula for computing $\pi(C(u;xy))$ (for an illustration of this and subsequent cases see Fig.5):

$$\pi(C(u;xy)) = \pi(u) + \pi(C(x;z_0z_1)) + \sum_{i=1}^{p-1} \pi(C(y;z_iz_{i+1}))$$
$$- \pi(C(x;z_0z_1) \cap C(y;z_1z_2))$$

$$-\sum_{i=2}^{p-1} \pi(C(y; z_{i-1}z_i) \cap C(y; z_i z_{i+1})). \qquad (1)$$

(Notice that in (1) the weight of y is added $p-1$ times and is substracted $p-2$ times.) Now, if z_0 does not exist, then simply replace in (1) the cone $C(x; z_0 z_1)$ by the degenerate cone $C(x; z_1 z_1)$ if x is adjacent to z_1 and by $\{x\}$ otherwise. On the other hand, if some consecutive vertices z_k and z_{k+1} are not adjacent, then replace in (1) the cone $C(y; z_k z_{k+1})$ by the degenerate cone $C(y; z_{k+1} z_{k+1})$. In particular, replace $C(y; z_{p-1} z_p)$ by $C(y; z_p z_p)$ if $k = p-1$.

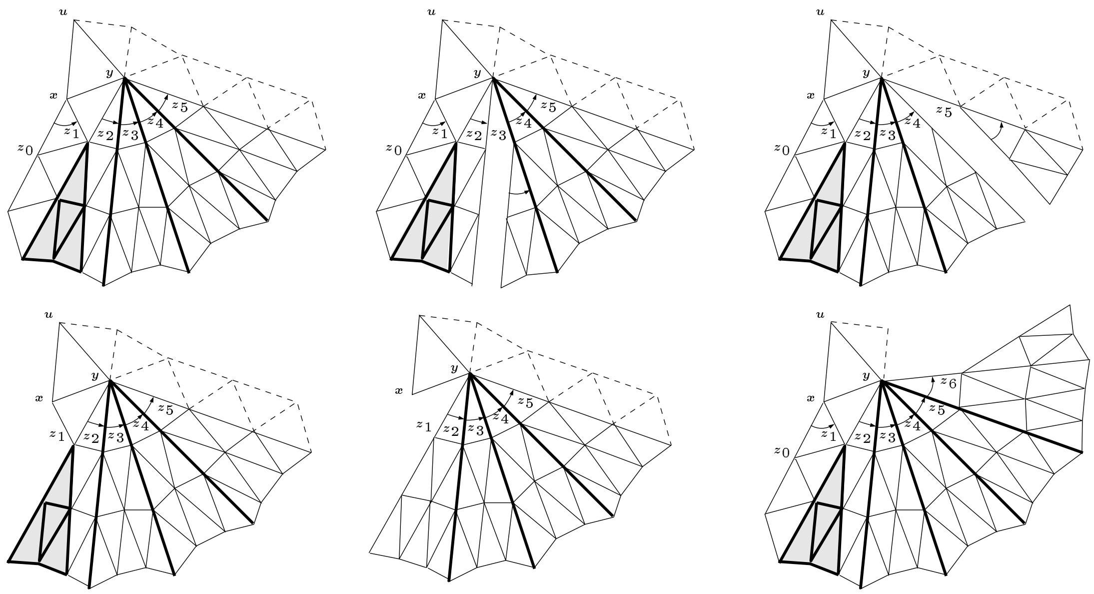


Fig. 5

Below we will describe how to organize the computation on G so that each time we wish to compute $\pi(C(u; xy))$, the weights of all cones and rays occurring in the right-hand side of (1) have been already computed. For this, pick a vertex c on the outer face of G and perform the levelling of the graph G in the following way: for an integer i define ith *level* L_i to be the subgraph induced by all vertices of G located at distance i from c, see Fig.6. We call an edge uv of G *horizontal* if both u and v belong to the same level and *vertical* otherwise.

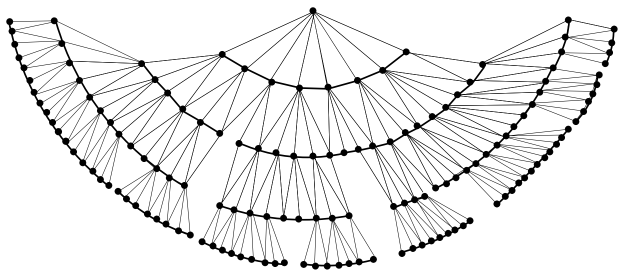


Fig. 6: levelling of a trigraph

Lemma 3. *Every connected component in each level L_i is a path.*

Proof. Notice that the union of the levels L_j $(j < i)$ is the ball $B_{i-1}(c)$, therefore it is a convex subset of G. Since G is K_4-free, this implies that every vertex of L_i is adjacent to at most two consecutive vertices in the previous level L_{i-1}. From triangle condition we know that any two adjacent vertices x, y of L_i have a common neighbor u in L_{i-1}. Since G is K_4-free, this common neighbor is necessarily unique.

Now, assume by way of contradiction that L_i contains three pairwise adjacent vertices x, y, z. Let z', y', x' be the common neighbors in L_{i-1} of x, y, of x, z, and of y, z, respectively. Convexity of $B_{i-1}(c)$ and the fact that G is K_4-free imply that x', y', z' are distinct and pairwise adjacent. Since x and y have already two neighbors in L_{i-1}, we conclude that the vertices x, y', x', y induce a 4-cycle, which is impossible. Finally, suppose that L_i contains a vertex x adjacent to three other vertices y, z, v. Now, if we consider the common neighbors y', z', v' in L_{i-1} of, respectively, y and x, z and x, and v and x, then each two of them either coincide or are adjacent. If y', z', v' are pairwise distinct, then together with x they will form a K_4. On the other hand, if all these vertices coincide, then together with x, y, z, v they induce a forbidden $K_{1,1,3}$. Finally, if $y' = z' \neq v'$, then x, y, z, y', v' induce a $K_{1,1,3}$. Hence, every vertex of L_i has degree 1 or 2 and L_i does not contain triangles. Thus every connected component of L_i is a path or a cycle. Since the base-point b belongs to the outer face, one can easily see that the second case is impossible, thus L_i consists solely of paths. $\square$

With respect to the levelling of G, we present the following classification of cones of G. A cone $C(u; xy)$ is called a D-*cone* if $u \in L_{i-1}$ and $x, y \in L_i$, and a RD-*cone* if $u, x \in L_{i-1}$, $y \in L_i$, and x is right from u on L_{i-1}. Analogously one can define the LD-*cones*, the U-*cones*, the LU-*cones*, and the RU-*cones.* Call a {D,RD,LD}-cone a *downward cone* and a

{U,RU,LU}-cone an *upward cone*. Clearly, every cone of G is of one of these six types; for illustrations see Fig.7.

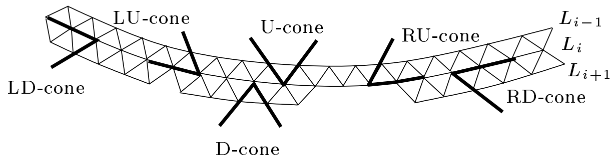


Fig. 7

The computation of the weights of cones is performed in the following way. First, we sweep G level by level in decreasing order of their distances to c (upward) and compute the weights of downward cones. In order to compute the LD-cones with apices in the ith level, L_i is swept from left to the right, while to compute the analogous RD-cones, L_i is swept from right to the left (the D-cones can be computed at each of these traversals). Since every vertex of L_{i+1} has one or two adjacent neighbors in L_i, one can easily see that every cone used in the computation of the weight of some downward cone with the apex at L_i may occur at most four times at the right-hand side of (1). Therefore the weights of the downwards cones with apices at L_i can be computed in time proportional to the number of edges in the subgraph induced by $L_i \cup L_{i+1}$, whence the overall computation of weights of downward cones is linear.

Now, to compute the weights of upward cones, we sweep the levels of the graph G in increasing order of their distances to c (downward). At stage i, we traverse the level L_i from right to left, and for every vertex $y \in L_i$, we compute the weights of all upward cones having y as the left end-vertex of their generator, i.e. of all cones $C(u; xy)$ such that the half-edge $\overrightarrow{xy}$ occurs in the counterclockwise traversal of the inner face (u, x, y). However, computing $\pi(C(u; xy))$ directly via (1) would not yield a linear time algorithm because every cone with apex y appears in the right-hand side of this formula for all upward cones $C(u; xy)$ except a constant number. Instead, we proceed in the following way. Let $z_0, z_1, \ldots, z_{p-1}, z_p$ be the neighbors of y in G ordered in the counterclockwise order, where $z_0, z_p \in L_{i-1}, z_1, z_{p-1} \in L_i$ and the remaining neighbors are in L_{i+1}. First, applying (1) we compute the weight of the rightmost upward cone $C(z_{p-1}; z_p y)$. Then we successively update this weight by turning around the vertex y in clockwise order. Assume for example that we wish to compute the weights of the upward cones $C(z_{i-1}; z_i y)$ $(i = 2, \ldots, p-1)$. For this, notice that the symmetric difference between two consecutive cones $C(z_i; z_{i+1}y)$ and $C(z_{i-1}; z_i y)$ consists of two cones $C(y; z_j z_{j+1})$ and $C(z_i; zz_{i+1})$, where z is the common neighbor of z_i and z_{i+1} different from y (if z does not exist, then the second cone is the degenerated cone $C(z_i; z_{i+1}z_{i+1})$). As we will establish below, one can suppose that the weights of these two cones have been already computed. Now, knowing $\pi(C(z_i; z_{i-1}y))$, the weight of the cone $C(z_{i+1}; z_i y)$ is obtained by setting

$$\pi(C(z_{i-1}; z_i y)) := \pi(C(z_i; z_{i+1}y)) + \pi(C(z_i; zz_{i+1})) - \pi(C(y; z_j z_{j+1}))$$

(for an illustration see Fig.8). Clearly, the complexity of performing these computations for a given vertex y is proportional to its degree, therefore the overall time of computing the upward cones is also linear.

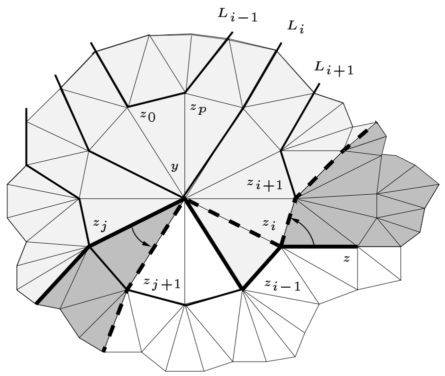


Fig. 8: $C(z_i; z_{i+1}y) \,\Delta\, C(z_{i-1}; z_i y) = C(y; z_j z_{j+1}) \cup C(z_i; zz_{i+1})$

The correctness of this algorithm follows from the following result.

Lemma 4. *If $C(u; xy)$ is the current cone, then the weights of cones arising at the right-hand side of (1) have been already computed.*

Proof. The basic ingredients of the proof are Lemma 3 and the following facts about $C(u; xy)$. First, from Lemma 2 we conclude that the cone $C(u; xy)$ is convex and that its rays are convex paths. Second, for every vertex $z \neq u$ of $C(u; xy)$ every shortest path between u and z intersects the generator $\{x, y\}$. As above, by $z_0, z_1, \ldots, z_p$ we denote the neighbors of y and/or x in $C(u; xy)$. From previous properties of cones, we conclude that neither of these vertices is adjacent to u. Now, suppose that the levelling of G has n levels and that u belongs to L_i. We proceed by induction on $n - i$ for downward cones and by induction on i for upward cones.

Case 1. $C(u;xy)$ is a downward cone.

If $x, y \in L_{i+1}$ (i.e., $C(u;xy)$ is a D-cone), however some z_j belongs to L_i, then z_j and u must be adjacent because they have a common neighbor outside the ball $B_i(c)$, which is impossible. So, assume without loss of generality that $x \in L_{i+1}$ and $y \in L_i$. Since u is not adjacent to z_0, we conclude that $z_0, z_1 \notin B_i(c)$, thus the weight $\pi(C(x; z_0z_1))$ is already known in view of induction hypothesis. As to the cones $C(y, z_jz_{j+1})$, assume by way of contradiction that some z_j belongs to the level L_{i-1}. Since z_j is not adjacent to u and $u, y \in L_i$, by triangle condition there exists a common neighbor $z \neq z_j$ of u and y one step closer to c. Since $z, z_j \in L_{j-1}$ and both these vertices are adjacent to y, the convexity of the ball $B_{i-1}(c)$ yields that z is adjacent to z_j. Then $z \in I(u, z_j) \subset C(u;xy)$, which is impossible.

Case 2. $C(u;xy)$ is an upward cone.

First assume that $x, y \in L_{i-1}$, and pick a cone from the right-hand side of (1), say the cone $C(y; z_jz_{j+1})$. If the vertices of its generator belong to the levels L_{i-1} and L_{i-2}, then, by the induction hypothesis, the weight of this cone is known. On the other hand, if one vertex of its generator belongs to L_{i+1} and another one to L_i or L_{i+1}, then $C(y; z_jz_{j+1})$ is a downward cone, therefore its weight has been computed at previous stage. The case when $C(u;xy)$ is a LU- or DU-cone is analogous subject to minor modifications. For example, if, say $y \in L_i$ and $x \in L_{i-1}$, then no cone $C(y; z_jz_{j+1})$ may have both z_j and z_{j+1} in L_{i-1} : the convexity of $B_{i-1}(c)$ then implies that x, y, z_j, z_{j+1} are pairwise adjacent and we get a K_4. In all other cases, $C(y; z_jz_{j+1})$ is either a downward or an upward cone whose weight has been already computed due to induction hypothesis. □

4.5. Computing the weights of half-planes. Let H_i, H'_i be a pair of complementary half-planes defined by the zone Z_i. Let uv and $u'v'$ be the boundary edges of Z_i so that u, v' are the end-vertices of P_i and v, u' are the end-vertices of P'_i. We will show how to compute $\pi(H_i)$ ($\pi(H'_i)$ can be computed analogously but using the vertex u'). Denote by $z_1 := v, \ldots, z_p$ the neighbors of the vertex u ordered clockwise. Then $z_2, \ldots, z_p$ are the neighbors of u lying in the half-plane H_i; see Fig.9. Now notice that H_i is the union of the non-degenerated cones $C(u; z_jz_{j+1})$ ($j = 2, \ldots p-1$) and of the degenerated cone $C(u; z_pz_p)$. The u-rays defined by the edges $uz_3, \ldots, uz_{p-1}$, being the intersection of two consecutive cones, are counted twice.

Hence

$$\pi(H_i) = \sum_{j=2}^{p} \pi(C(u; z_j z_{j+1})) - \sum_{j=2}^{p-1} \pi(C(u; z_j z_{j+1}) \cap C(u; z_{j+1} z_{j+2})),$$

where $C(u; z_p z_{p+1})$ stands for the degenerated cone $C(u; z_p z_p)$. This shows that the weight of H_i can be computed in time proportional to the degree of the vertex u. The vertices u and v are separated by two pairs of complementary half-planes, therefore u will be involved in computing the weights of two half-planes only. This proves that the weights of the half-planes if G can be computed in time proportional to the sum of degrees of vertices of ∂G, i.e., in linear time.

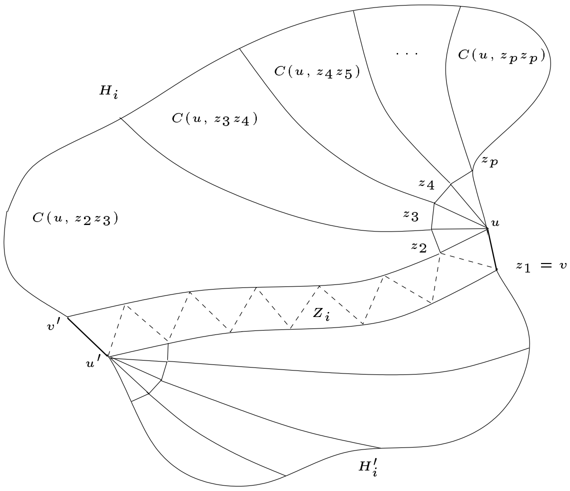


Fig. 9

4.6. Complexity and analysis of the algorithm MEDIAN SET. While describing in details steps 1-4 of the algorithm, we established that the complexity of each of these steps is linear. When the weights of complementary half-planes H_i, H_i' are computed, then they are broadcasted to all edges of the zip Z_i. Now, given an edge $uv \in Z_i$, the weights of $W(u,v)$ and $W(v,u)$ can be found in constant time as noticed in step 5 and illustrated in Fig. 3. Hence step 5 needs $O(|E|)$ operations, the same order as the steps 6 and 7. Since $|E| \leq 3|V| - 2$ because G is planar, we conclude that the complexity of the algorithm MEDIAN SET is $O(|V|)$. This algorithm can be modified (even simplified) in order to compute the median sets of squaregraphs (we skip the straightforward details). Concluding, we obtain the following result:

Theorem 1. *For every weight function π defined on vertices of a trigraph or a squaregraph $G = (V, E)$, the median set* $\mathrm{Med}(\pi)$ *can be computed in linear time $O(|V|)$.*

References

[1] S. Alstrup, J. Holm, and M. Thorup, Maintaining center and median in dynamic trees, in: SWAT'00, 7th Scandinavian Workshop on Algorithm Theory, 2000, 46-56.

[2] G. Antonoiu, P. K. Srimani, A self-stabilizing distributed algorithm to find the median of a tree graph, J. Computer and System Sciences 58(1) (1999) 215-221.

[3] B. Aronov, M. van Kreveld, R. van Oostrum, and K. Varadarajan, Facility location on terrains, Discrete & Computational Geometry (to appear).

[4] V. Auletta, D. Parente, and G. Persiano, Dynamic and static algorithms for optimal placement of resources in a tree, Theoretical Computer Science 165 (1996) 441-461.

[5] H.-J. Bandelt and J.P. Barthélémy, Medians in median graphs, Discrete Applied Mathematics 8 (1984) 131-142.

[6] H.-J. Bandelt and V. Chepoi, Decomposition and l_1–embedding of weakly median graphs, European J. Combinatorics 21 (2000) 701-714.

[7] H.-J. Bandelt and V. Chepoi, Graphs with connected medians, SIAM J. Discrete Mathematics 15 (2002) 268-282.

[8] E. Barcucci, A. Del Lungo, M. Nivat and R. Pinzani, Medians of polyominoes: a property for the reconstruction, Int. J. Imaging Systems and Technology 8 (1998) 69-77.

[9] J.P. Barthélemy and B. Monjardet, The median procedure in cluster analysis and social choice theory, Mathematical Social Sciences 1 (1981) 235-268.

[10] O. Baues and N. Peyerimhoff, Curvature and geometry of tessellating plane graphs, Discrete & Computational Geometry 25 (2001) 141-159.

[11] M. Bernstein, V. de Silva, J.C. Langford, and J.B. Tenenbaum, Graph approximations to geodesics on embedded manifolds, manuscript (2000) 24pp.

[12] P. Bose, A. Maheshwari and P. Morin, Fast approximations for sums of distances, clustering and the Fermat-Weber problem, Computational Geometry: Theory and Applications, 2001 (in print).

[13] M.R. Bridson and A. Haefliger, Metric Spaces of Non-Positive Curvature (Springer, Berlin, 1999).

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

[15] V. Chepoi and F. Dragan, Computing the median point of a simple rectilinear polygon, Information Processing Letters 49 (1994) 281-285.

[16] V. Chepoi, F. Dragan, and Y. Vaxès, Center and diameter problems in plane triangulations and quadrangulations, in: SODA'02, 13th ACM-SIAM Annual Symposium on Discrete Algorithms, 2002, 346-355.

[17] A. Daurat, A. Del Lungo and M. Nivat, The medians of discrete sets according to some linear metrics, Discrete & Computational Geometry 23 (2000) 465-483.

[18] M de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, Computational Geometry. Algorithms and Applications (Springer, Berlin, 1997).

[19] A. Del Lungo, M. Nivat, R. Pinzani, and L. Sorry, The medians of discrete sets, Information Processing Letters 65 (1998) 293-299.

[20] A.J. Goldman, Optimal center location in simple networks, Transportation Science 5 (1971), 212-221.

[21] A.J. Goldman C.J. Witzgall, A localization theorem for optimal facility placement, Transportation Science 4 (1970) 406-409.

[22] Y. Métivier and N. Saheb, Medians and centers of polyominos, Information Processing Letters 57 (1996) 175-181.

[23] F.R. McMorris and R.C. Powers, The median procedure in a formal theory of consensus, SIAM J. Discete Mathematics 8 (1995) 507–516.

[24] B.C. Tansel, R.L. Francis, and T.J. Lowe, Location on networks: a survey I, II, Management Science 29 (1983) 482–511.

[25] J.B. Tenenbaum, V. de Silva, and J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319-2323.

[26] M.L.J. van de Vel, Theory of Convex Structures (North-Holland, Amsterdam, 1993).

[27] B. Zelinka, Medians and peripherians of trees, Archivum Mathematicum 4 (1968) 87-95.