Locating Median Paths on Connected Outerplanar Graphs

I. Lari* F. Ricca* A. Scozzari † R. I. Becker‡

Abstract

During the last two decades, there has been a growing interest in locating extensive facilities, such as paths, on networks. In this paper we study the median path problem without restrictions on its length on the class of connected outerplanar graphs with equal weights assigned to the edges and nonnegative weights associated to the vertices. We provide a $O(kn)$ time algorithm, where n is the number of vertices of the graph G and k is the number of blocks in G. As a byproduct, we provide a linear time algorithm to find a median path without restrictions on its length between two fixed vertices in a biconnected outerplanar graph.

Keywords: Path location, path median, biconnected outerplanar graphs.

1 Introduction

During the last two decades, there has been a growing interest in the developing of location models on networks, with particular attention to the location of extensive facilities, such as paths or trees. In almost all cases, the criteria used are the *minsum* criterion, according to which the sum of the distances from all the vertices of the network to the facility is minimized, and the *minimax* criterion, that is, the distance from the facility to the farthest vertex in the network is minimized. The present paper investigates the problem of locating a path-shaped facility with the minsum criterion without restrictions on the length of the path. Examples of such a problem include the location of pipelines, evacuation routes, mass transit routes or routing a highway through a road network, and public transit lines. An optimal path for this problem is also referred to as a *median path*, thus, in the rest of the paper, we will refer to the problem under study as the Median Path Problem (MPP).

In the literature MPP was widely studied when there is a restriction on the length of the path. On general networks, this problem is NP-complete [5, 12, 3]. In particular, in [5] it is shown that it is NP-Complete on planar graphs with vertex degree less than or equal to 5, while [3] provides the same result on rectangular grid graphs. In [12] it is shown that the median path problem with length at most equal to a given constant, is NP-hard on outerplanar graphs, but in [7] it is actually shown that the same problem is NP-hard even on the class of cactus graphs. Nevertheless, [12] provides a pseudo-polynomial time algorithm for the solution of MPP with restricted length on series-parallel graphs. If the graph is unweighted, the above algorithm has an overall time complexity of $O(n^7 \log n)$, where n is the number of vertices of the graph.

For MPP with restricted length on general networks different directions were investigated: in [7] a metaheuristic approach was presented, while [2] suggests a branch-and-cut algorithm. Finally, a number of papers have investigated the problem of locating median paths with restricted length on trees and efficient polynomial time algorithms were provided (see, e.g., [1, 4, 5, 9, 11]).

Since also MPP (without restrictions on the length of the path) is NP-hard on general networks [5], this problem was mainly studied on trees, too [10, 11, 13], while, to the best of our knowledge, it has not been studied yet on networks with cycles.

In this paper we study MPP on the class of outerplanar graphs. Notice that, if we consider a

*Università di Roma "La Sapienza", Dip. Statistica, Probabilità e Statistiche Applicate.

†Università di Roma "La Sapienza", Dip. Matematica per le Decisioni Economiche, Finanziarie ed Assicurative.

‡University of Cape Town, Dep. of Mathematics and Applied Mathematics.

biconnected outerplanar graph G, the solution is trivial, since a median path without restrictions on the length is simply given by the path passing through all the vertices on the outercycle of G. On the other hand, this is not true if we consider the case of finding a median path between two fixed end vertices in a biconnected outerplanar graph. Actually, this will turn out to be a special case of our more general problem for which we will provide an algorithm linear in the number of vertices of the graph. In this paper we consider the more general class of *connected* outerplanar graphs (or, simply, outerplanar graphs). Figure 1 shows an example of a connected outerplanar graph that is not biconnected. In particular, we focus our attention on the case in which equal weights are assigned to the edges of G, while nonnegative weights are associated to the vertices of G. We show that for a given graph G with n vertices we can suitably decompose G into k components and represent it by a tree, $\mathcal{T}$, that we call *representation tree* of G. For solving MPP in an outerplanar graph G we provide an algorithm with complexity $O(kn)$.

The remainder of this paper is organized as follows: Section 2 provides some notation, definitions and some basic properties; Section 3 describes the preprocessing phase that is needed in order to compute the basic quantities associated to the vertices and the edges of the graph, while Section 4 provides the algorithm for solving MPP in an outerplanar graph. Section 5 provides some concluding remarks and extensions.

2 Basic properties and definitions

Let $G = (V, E)$ be an outerplanar graph, where V is the vertex set, $|V| = n$, and E is the edge set. Here G is assumed to be finite, undirected, connected and without multiple edges or self-loops. Suppose that a weight equal to one is assigned to each edge, and a nonnegative weight w_v is assigned to each vertex $v \in V$. For any two vertices u and v, we define the distance $d(u, v)$ between u and v as the number of edges (length) of the shortest path between them in G. Given any path P, the distance from u to P, $d(u, P)$, is the minimum among the distances from u to the vertices in P. Given a path P, the *distsum* $D(P)$ of P is the sum of the weighted distances from all the vertices of G to P, that is, $D(P) = \sum_{u \in V} w_u d(u, P)$.

In this paper we study the following MPP:

Given a connected outerplanar graph G, find a path P^* in G such that $D(P^*)$ is a minimum w.r.t. all the possible paths P in G.

Given a graph G, a vertex v is called *cut vertex* if removing v and all edges incident to it disconnects G. A *bridge* is an edge of G whose removal increases the number of connected components of G [6]. Notice that the end vertices of a bridge are cut vertices of G. A *block* is a maximal subgraph of G with no cut vertices and a *face* is a chordless cycle of G. Notice that a block may correspond to a face, or it may be composed by a set of faces.

An outerplanar graph G can be decomposed into blocks and bridges and it can be represented by a tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$, where $V_{\mathcal{T}}$ is the set of blocks and bridges of G. In the rest of the paper we will denote by B both a block or bridge of G and the corresponding vertex of $\mathcal{T}$. There is an edge between two vertices of $\mathcal{T}$ if they share a cut vertex in G (see, Figure 1). We call $\mathcal{T}$ the *representation tree* of G. We root $\mathcal{T}$ at any block H and we denote by $\mathcal{T}_H$ the resulting rooted tree. Following the usual notation in graph theory [6], given $B \neq H$, the subtree of $\mathcal{T}_H$ rooted at B is denoted by $\mathcal{T}_B$ and the set of its vertices by $V(\mathcal{T}_B)$. A block of G that corresponds to a leaf of $\mathcal{T}$ will be called *leaf block*. With respect to the representation tree $\mathcal{T}$, we can state some properties that characterize an optimal path P^* in an outerplanar graph.

Property 1 The end vertices of an optimal path P^* do not necessarily belong to a leaf block of $\mathcal{T}$ (see, for example Figure 2).

Property 2 Suppose that an optimal path P^* ends in a block or a bridge B of G. Then, all the

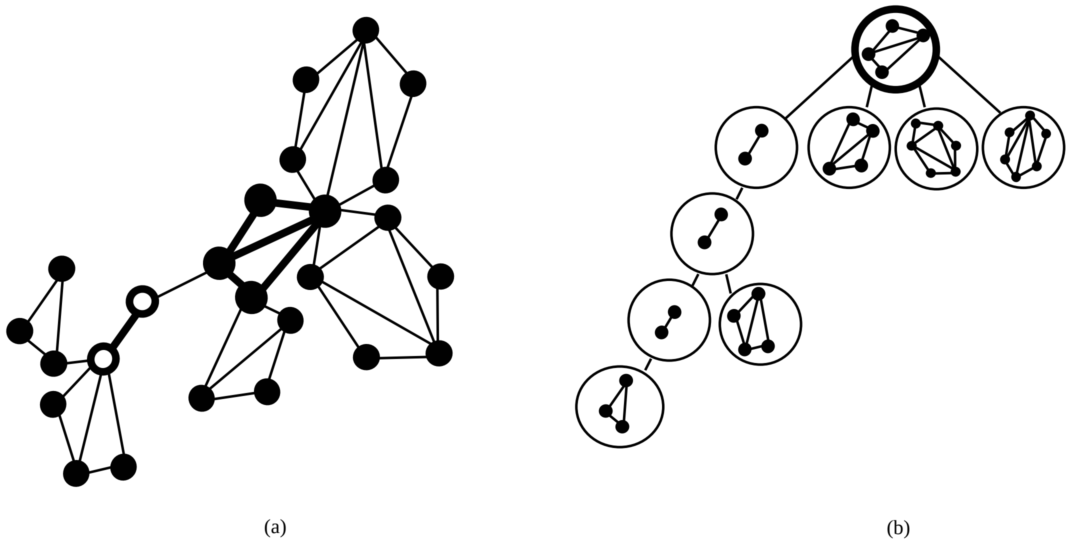

Figure 1: (a) An outerplanar graph: one of its blocks is highlighted in boldface, while one of its bridges is given by the edge with white end vertices. Both these end vertices are examples of cut vertices. (b) The representation tree of G rooted at the block in boldface.

vertices of B belong to P^*. In particular, if B is a bridge, it must contain a vertex having degree 1, and, thus, it must be a leaf of the representation tree $\mathcal{T}$.

Notice that, after Property 2, an optimal path P^* never ends in a non-leaf bridge.
This property is straightforward since, if some vertices of a block B were not included in P^*, then, another path P could be obtained by including such vertices, thus obtaining $D(P) < D(P^*)$, a contradiction. The property trivially holds also for bridges.

After Properties 1 and 2, we know that, if G has more than one block, an optimal path P^* of G starting at a block H and ending in a block B must include all the vertices of H and B. In addition, due to the structure of G, P^* must extend out of H passing through a cut vertex. Thus, given a graph G, the idea of the algorithm is the following. We root the representation tree $\mathcal{T}$ at any block H and we visit $\mathcal{T}_H$ top down, block by block in a BFS. order. At each block, the visit proceeds face by face inside the block. Let u be a vertex belonging to a face f in a block B; then, among all the paths starting from H and ending in u, we search for the one that minimizes the *distsum*. We call such a path *best path* from H to u and we denote it and its *distsum* by $P_H(u)$ and $D_H(u)$, respectively. To simplify our notation, when this does not cause any confusion, we refer to $P_H(u)$ and $D_H(u)$ simply by $P(u)$ and $D(u)$, respectively. In order to find P^*, among

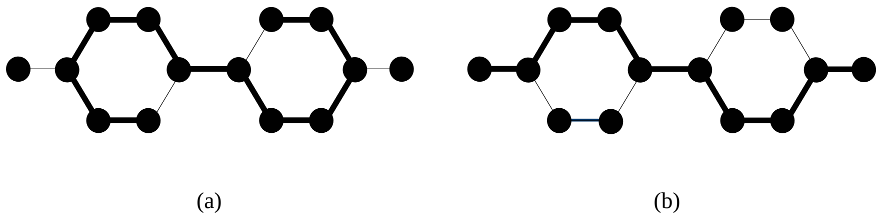

Figure 2: An outerplanar graph G with two faces and three bridges. Here the leaf blocks of the corresponding representation tree are the left- and right-most bridges G. Suppose that each vertex has weight equal to 1: the path in (a) has *distsum* equal to 2, while the path in (b) has *distsum* equal to 4.

all the best paths $P_H(u)$ found for every possible H and u, we select the one with the minimum *distsum*.

Consider $\mathcal{T}_H$ and any block $B \neq H$ and let $V(B)$ be the set of vertices of B. Denote by c_B the cut vertex that B shares with its parent in $\mathcal{T}_H$. We define c_B as the *source* of B. Since B is a biconnected outerplanar graph and all its vertices lie on the outercycle of B, it is always possible to number clockwise the vertices of B in an increasing order such that the label equal to $|V(B)|$ is assigned to c_B (see Figure 3). Suppose that F_B is the number of faces in B. It is also possible to assign a number f, $f = 1, ..., F_B$, to the faces of B such that $c_B \in F_B$ and each face f is adjacent to (i.e., has a chord in common with) exactly one face with a number grater than f [8]. We will denote by (r_f, t_f) a chord separating face $f \neq F_B$ from the unique face f' adjacent to f with $f' > f$. To be consistent with the numbering of the vertices, we assume $t_f < r_f$, and for any edge (u, v) in face f, we suppose that $v < u$ (see Figure 3). With respect to face f, we denote by $V(f)$ and $E(f)$ the set of vertices and the set of edges in f, respectively. Finally, we define the set $V(r_f, t_f)$ as the set of vertices v in $V(B)$ with number $t_f \leq v \leq r_f$. With respect to a given $\mathcal{T}_H$, for a chord (u, v) belonging to face f of a block $B \neq H$, we denote by $\Pi(u, v)$ the unique path in B connecting u to v that passes through all the vertices in $V(u, v)$. For example, in Figure 3 we have $\Pi(8, 5) = \{(5, 6)(6, 7)(7, 8)\}$.

Property 3 Given a block B in G, consider the clockwise numbering of the vertices of B introduced above. For any pair of vertices s and t in B, a path from s to t that minimizes the *distsum* never passes through a chord (u, v) with $s \leq v \leq t$ and $s \leq u \leq t$.

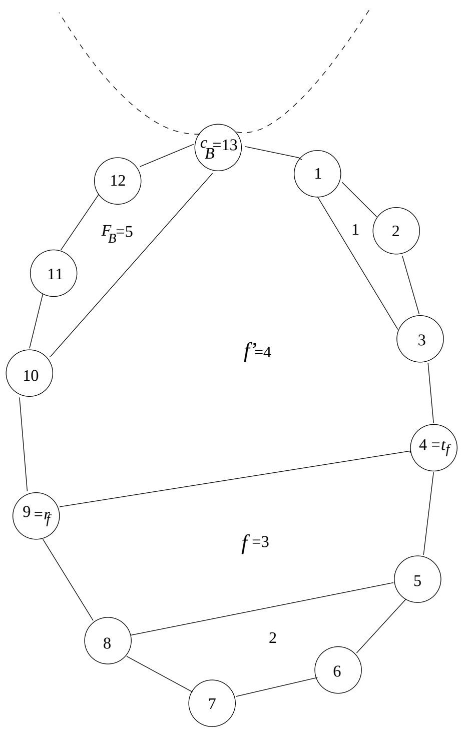

Figure 3: Notation in an outerplanar graph.

3 The preprocessing phase

In order to provide an algorithm for the MPP in G, we need a preprocessing phase in which, for any given block H of G we visit the tree $\mathcal{T}_H$ bottom up, level-by-level, and compute some quantities associated both to its edges and its vertices. Nevertheless, unlike the preprocessing usually implemented for median path location problems on trees (see for example [4, 10]), here the particular structure of each vertex of $\mathcal{T}_H$ (i.e., a block of G) requires a separated procedure to compute the necessary quantities. We first focus on this point, before giving the detailed description of the whole preprocessing.

Given a vertex of $\mathcal{T}_H$, which corresponds to a block B, and its source cut vertex c_B, we need to compute the following quantities:

S_{c_B}, the sum of the weighted distances to c_B from all the vertices of G belonging to the blocks in $\mathcal{T}_B$;

W_{c_B}, the sum of the weights of the vertices of G belonging to the blocks in $\mathcal{T}_B$.

In order to compute S_{c_B} and W_{c_B}, we visit the faces of B from $f = 1$ to $f = F_B - 1$. For a given face f we compute the following quantities:

W_{t_f}, the sum of the weights of the vertices of $V(\mathcal{T}_B)\backslash[V(B)\backslash V(r_f, t_f)]$ that are closer to t_f than to r_f; such vertices are said to be *assigned to* t_f in f.

W_{r_f}, the sum of the weights of the vertices of $V(\mathcal{T}_B)\backslash[V(B)\backslash V(r_f, t_f)]$ that are closer to r_f than to t_f; such vertices are said to be *assigned to* r_f in f.

When in f for a vertex v we have $d(v, r_f) = d(v, t_f)$, this vertex cannot be definitively assigned to only one between r_f and t_f. We refer to such vertices as *unassigned* in f, and denote by $W_{r_f t_f}$ the sum of their weights.

Similarly, we compute S_{r_f} (S_{t_f}) as the sum of the weighted distances to r_f (t_f) from all the vertices assigned in f to r_f (t_f), while the sum of the weighted distances from all the unassigned vertices v in f - computed either with respect to $d(v, r_f)$ or $d(v, t_f)$ - is denoted by $S_{r_f t_f}$.

For a given $\mathcal{T}_H$, the blocks are analyzed following a bottom up visit of $\mathcal{T}_H$. The quantities associated to r_f and t_f are computed during a clockwise visit of the outercycle of each block B, that corresponds to the visit of the faces of B from $f = 1$ to $f = F_B - 1$ (see Figure 3). Then, in F_B, we compute the quantities associated to c_B.

Notice that in our algorithm we need only the quantities associated to the cut vertices of G and to pairs of vertices in V that correspond to chords of G. Nevertheless, we compute the same quantities for all the vertices $u \in V$ and all the edges $(u, v) \in E$. At the beginning we set $W_u = w_u$ and $S_u = 0$, for all $u \in V$ and $W_{uv} = S_{uv} = 0$, for all $(u, v) \in E$.

In our preprocessing, in each face f we need to compute an additional quantity associated to the chord (r_f, t_f), which is denoted by $Sav(r_f, t_f)$. This quantity is necessary when, during the algorithm, we visit the unique face f' that is adjacent to f with $f' > f$. $Sav(r_f, t_f)$ is the *saving* in the *distsum* that can be obtained in a path from H to $u \in V(f)$ containing both r_f and t_f, when the subpath $\Pi(r_f, t_f)$ is included instead of the chord (r_f, t_f). Also for these quantities we compute $Sav(u, v)$ for all $(u, v) \in E$. If (u, v) is not a chord, we have no saving at all. Thus, at the beginning we set $Sav(u, v) = 0$, for all (u, v) in G and, during the preprocessing, we update only the quantities corresponding to the chords of G.
For a block B in $\mathcal{T}_H$, the following procedure is repeatedly applied to each face from $f = 1$ to $f = F_B - 1$ in order to compute W_{t_f}, W_{r_f}, $W_{r_f t_f}$, S_{t_f}, S_{r_f}, $S_{r_f t_f}$, $Sav(r_f, t_f)$.

algorithm FACE(f)

input: A face f, with vertices r_f, t_f, weights W_u and S_u $\forall u$ in $V(f)$, weights W_{uv} and S_{uv} $\forall (u,v)$ in $E(f)$.
output: Updated values for W_{t_f}, W_{r_f}, $W_{r_f t_f}$, S_{t_f}, S_{r_f}, $S_{r_f t_f}$, $Sav(r_f, t_f)$.

```
begin
    $v = t_f$
    Let $(u, v)$ be the unique edge in $E(f)$ such that $v < u$
    while $u < r_f$
        Consider the quantities $W_u$, $S_u$, $W_{uv}$ and $S_{uv}$
        if $(d(u, t_f) < d(u, r_f))$
            $W_{t_f} = W_{t_f} + W_u + W_{uv}$
            $S_{t_f} = S_{t_f} + (S_u + W_u d(u, t_f)) + (S_{uv} + W_{uv} d(v, t_f))$
        else if $(d(u, t_f) = d(u, r_f))$
            $W_{t_f} = W_{t_f} + W_{uv}$
            $W_{r_f t_f} = W_{r_f t_f} + W_u$
            $S_{t_f} = S_{t_f} + (S_{uv} + W_{uv} d(v, t_f))$
            $S_{r_f t_f} = S_{r_f t_f} + (S_u + W_u d(u, t_f))$
        else $[d(u, t_f) > d(u, r_f)]$
                if $(d(u, r_f) = d(v, t_f))$
                    $W_{r_f} = W_{r_f} + W_u$
                    $W_{r_f t_f} = W_{r_f t_f} + W_{uv}$
                    $S_{r_f} = S_{r_f} + (S_u + W_u d(u, r_f))$
                    $S_{r_f t_f} = S_{r_f t_f} + (S_{uv} + W_{uv} d(u, r_f))$
                else
                    $W_{r_f} = W_{r_f} + W_u + W_{uv}$
                    $S_{r_f} = S_{r_f} + (S_u + W_u d(u, r_f)) + (S_{uv} + W_{uv} d(u, r_f))$
                end if
        end if
        $v = u$
        Let $(u, v)$ be the unique edge in $E(f)$ such that $v < u$
    end while
    $u = r_f$
    $W_{r_f} = W_{r_f} + W_{uv}$
    $S_{r_f} = S_{r_f} + S_{uv}$
    $Sav(r_f, t_f) = \sum_{i \in V(f)} W_i \min\{d(i, r_f), d(i, t_f)\} +$
        $+ \sum_{(i,j) \in E(f)} [W_{ij} \min\{d(i, r_f), d(j, t_f)\}] + \sum_{(i,j) \in E(f)} Sav(i, j)$
end
```

To compute W_B and S_B in $f = F_B$ we apply the following formulas:

$W_{c_B} = W_{c_B} + \sum_{u \in V(f) | u \neq c_B} W_u$;

$S_{c_B} = S_{c_B} + \sum_{u \in V(f)} (S_u + W_u d(u, c_B)) + \sum_{(u,v) \in E(f)} (S_{uv} + W_{uv} \min\{d(u, c_B), d(v, c_B)\})$.

The pseudo-code for the whole preprocessing on $\mathcal{T}_H$ is reported below.

```
algorithm PREPROCESSING($\mathcal{T}_H$)

input: A weighted outerplanar graph $G$ and its representation tree rooted at block $H$, $\mathcal{T}_H$.
output: Updated values for $W_u$, $S_u$, for all $u$ in $V$; $W_{uv}$, $S_{uv}$, and $Sav(u,v)$, for all $(u,v)$ in $E$.

begin
    Visit $\mathcal{T}_H$ bottom up level-by-level.
    for each block $B \neq H$
        Starting from $c_B$, number clockwise the vertices of $B$ in an increasing order
        Number the faces of $B$ from 1 to $F_B$ such that $c_B \in F_B$ and each face $f$ is
        adjacent to exactly one $f' > f$
        for $f = 1, \ldots, F_B - 1$
            FACE($f$)
        end for
        $f = F_B$
        $W_{c_B} = W_{c_B} + \sum_{u \in V(f) | u \neq c_B} W_u$
        $S_{c_B} = S_{c_B} + \sum_{u \in V(f)} (S_u + W_u d(u, c_B)) + \sum_{(u,v) \in E(f)} (S_{uv} + W_{uv} \min\{d(u, c_B), d(v, c_B)\})$
    end for
end
```

The following proposition states the correctness of the above procedure.

Proposition 1 *For a given $\mathcal{T}_H$ and B in $\mathcal{T}_H$, let f' be a face and $f < f'$ be adjacent to f' in B. Consider a vertex $v \in V(r_f, t_f)$, $v \neq r_f, t_f$, then one of the following holds (see Figure 4):*

1. *v is assigned to r_f (t_f) in f. In this case, if r_f (t_f) is assigned in f' to some vertex (r'_f or t'_f), then v is assigned to the same vertex as r_f (t_f) in f'. Otherwise, if r_f (t_f) is unassigned in f', v is unassigned in f' as well;*

2. *v is unassigned in f. In this case, in f', v is assigned either to $r_{f'}$, or to $t_{f'}$, according to $d(r_f, r_{f'}) < d(t_f, t_{f'})$, or $d(r_f, r_{f'}) > d(t_f, t_{f'})$, respectively. If $d(r_f, r_{f'}) = d(t_f, t_{f'})$ v remains unassigned also in f'.*

Proof. To prove the first point, we notice that any path from v to $r_{f'}$, not containing r_f, must contain t_f. Since r_f and t_f are adjacent, we have $|d(t_f, r_{f'}) - d(r_f, r_{f'})| \leq 1$. In addition, by hypothesis, v is assigned to r_f and, thus, $d(v, t_f) = d(v, r_f) + 1$. By the Bellman optimality principle, there exists a shortest path from v to $r_{f'}$ containing r_f such that

$$d(v, r_{f'}) = d(v, r_f) + d(r_f, r_{f'}).$$

Similarly, there exists a shortest path from v to $t_{f'}$ containing r_f such that

$$d(v, t_{f'}) = d(v, r_f) + d(r_f, t_{f'}).$$

Hence, v is assigned to $r_{f'}$, or to $t_{f'}$ or it is unassigned if $d(r_f, r_{f'}) < d(r_f, t_{f'})$, or $d(r_f, r_{f'}) > d(r_f, t_{f'})$ or $d(r_f, r_{f'}) = d(r_f, t_{f'})$, respectively.
The same proof holds when v is assigned to t_f in f.

We now prove point 2. By the Bellman optimality principle we have:

$$\begin{array}{rcl} d(v, r_{f'}) & = & \min\{d(v, r_f) + d(r_f, r_{f'}), d(v, t_f) + d(t_f, r_{f'})\} \\ d(v, t_{f'}) & = & \min\{d(v, r_f) + d(r_f, t_{f'}), d(v, t_f) + d(t_f, t_{f'})\}. \end{array} \quad (1)$$

that, under the hypothesis that v is unassigned in f, reduces to:

$$\begin{array}{rcl} d(v, r_{f'}) &=& \min\{d(r_f, r_{f'}), d(t_f, r_{f'})\} \\ d(v, t_{f'}) &=& \min\{d(r_f, t_{f'}), d(t_f, t_{f'})\}. \end{array} \tag{2}$$

since $d(v, r_f) = d(v, t_f)$. Considering that r_f and t_f are adjacent, we also have

$$d(t_f, r_{f'}) = d(r_f, t_{f'}) = min\{d(r_f, r_{f'}), d(t_f, t_{f'})\} + 1, \tag{3}$$

It is easy to check that if $d(r_f, r_{f'}) < d(t_f, t_{f'})$, v is assigned to $r_{f'}$; if $d(r_f, r_{f'}) > d(t_f, t_{f'})$ v is assigned to $t_{f'}$, while, if $d(r_f, r_{f'}) = d(t_f, t_{f'})$, v remains unassigned in face f'. □

Proposition 2 *The preprocessing phase applied to $\mathcal{T}_H$ runs in $O(n)$ time.*

Proof. When visiting a face f, it is possible to update in constant time the distances $d(u, t_f)$ and $d(u, r_f)$ of each vertex $u \in V(f)$. Actually, let $L_f = |V(f)|$, so that $L_f - 1$ is the length of the path from t_f to r_f in face f that does not pass through the chord (r_f, t_f). The procedure FACE(f) starts from t_f, and visits f vertex by vertex. For each vertex $u \in V(f)$ it updates in constant time the length of the two paths connecting u to r_f and to t_f, respectively, and not containing the chord (r_f, t_f). Denote this two lengths by $\delta(u, r_f)$ and $\delta(u, t_f)$, respectively. For any pair of adjacent vertices u and v in f, $u, v \neq r_f, t_f$, $v < u$, we have $\delta(u, t_f) = \delta(v, t_f) + 1$ and $\delta(u, r_f) = L_f - 1 - \delta(u, t_f)$. In the preprocessing each face f is visited once, and, on the basis of Proposition 1, all the necessary quantities in f are correctly computed in time $O(|V(f)|)$. Notice that for any given block B, the computation of S_{c_B} and W_{c_B} is done in constant time. Hence, for a given representation tree $\mathcal{T}_H$, the overall time complexity of the preprocessing phase is $O(n)$. □

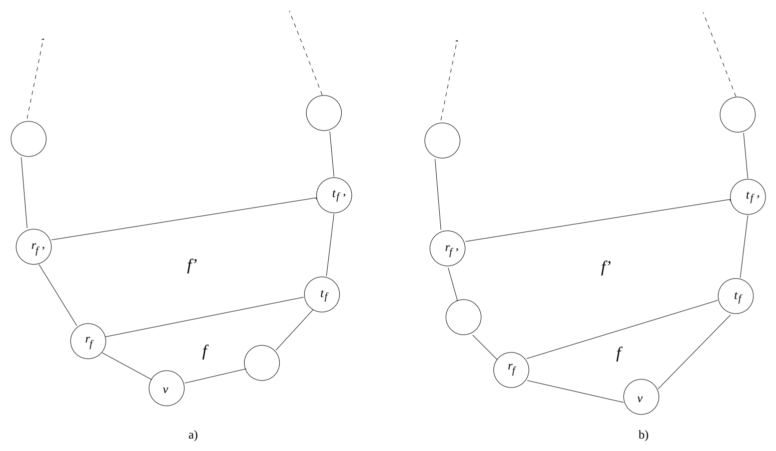

Figure 4: Examples of assigned and unassigned vertices. a) Vertex v is assigned to r_f in f and r_f is assigned to r'_f in f'; thus, v is assigned to r'_f in f', too. b) Vertex v is unassigned in f, but in f' it is assigned to t'_f.

4 The algorithm

The algorithm for finding a median path in an outerplanar graph G, roots the representation tree at each block H, and, for each vertex $u \notin V(H)$, finds the *distsum* of a best path from H to u. The algorithm first performs the preprocessing phase, then it computes the *distsum* D_H of any hamiltonian path of H by the following formula:

$$D_H = \sum_{u \in V(H)} S_u + \sum_{(u,v) \in E(H)} S_{uv} \tag{4}$$

The algorithm visits $\mathcal{T}_H$ in a BFS order and in each visited block B computes the *distsum* $D_H(u)$ of a best path from H to each vertex u of the block. If B is a bridge corresponding to the edge (c_B, u), the *distsum* of $P_H(u)$ is obtained by the following formula:

$$D_H(u) = D_H(c_B) - W_u. \tag{5}$$

If B is a biconnected block then, as in the preprocessing phase, the algorithm numbers the faces of B from 1 to F_B such that the source c_B belongs to face F_B. The visit of B starts from face F_B and proceeds face by face up to $f = 1$. In each face f the algorithm finds $D_H(u)$ for each vertex u of f. Notice that, for any given u, in face f only the *distsum* the subpath of $P_H(u)$ contained in face f must be computed. In order to evaluate the *distsum* of all possible such subpaths, the algorithm evaluates *clockwise* and *counterclockwise* subpaths w.r.t. f. When $f = F_B$ the subpath necessarily starts from the source c_B and it can reach vertex u visiting f either clockwise or counterclockwise. On the other hand, when $f \neq F_B$, it is necessary to distinguish four different cases that correspond to clockwise and counterclockwise subpaths starting either from r_f or from t_f.
Following the counterclockwise order, in each face f the algorithm assigns local labels $1, 2, \ldots, n_f$, with $n_f = |V(f)|$, to the vertices of f, so that, if $f = F_B$, vertex c_B has label 1 and, if $f \neq F_B$, vertex r_f has label 1 and vertex t_f has label n_f. Let $v_f : \{1, ..., n_f\} \to V(f)$ be the function that associates to each local label the corresponding vertex. For each vertex $u \in V(f)$ we consider the following two quantities that correspond to the minimum *distsum* for the different types of paths from H to u. For the sake of simplicity, in the rest of this section, we suppose that the root block H is fixed, so that in the notation we can drop the reference to it.

- $D^L(u)$ is the minimum *distsum* of a path from H to $u \in V(f)$ that visits f in a counterclockwise order (counterclockwise path for short);
- $D^R(u)$ is the minimum *distsum* of a path from H to $u \in V(f)$ that visits f in a clockwise order (clockwise path for short).

In addition, if $f \neq F_B$, the quantities $D^L(u)$ and $D^R(u)$ are computed on the basis of the following four quantities:

- $D^L_{r_f}(u)$ is the minimum *distsum* of a counterclockwise path from H to u containing r_f but not t_f;
- $D^L_{r_f,t_f}(u)$ is the minimum *distsum* of a counterclockwise path from H to u containing both r_f and t_f;
- $D^R_{t_f}(u)$ is the minimum *distsum* of a clockwise path from H to u containing t_f but not r_f;
- $D^R_{r_f,t_f}(u)$ is the minimum *distsum* of a clockwise path from H to u containing both r_f and t_f.

Then, w.r.t. vertex $u \in V(f)$ we have:

$$D(u) = \min\{D^L(u), D^R(u)\}. \tag{6}$$

and, in particular, when $f \neq F_B$, we have:

$$\begin{array}{l} D^L(u) = \min\{D^L_{r_f}(u), D^L_{r_f,t_f}(u)\} \\ D^R(u) = \min\{D^R_{t_f}(u), D^R_{r_f,t_f}(u)\} \end{array} \tag{7}$$

W.r.t. the local labels of the vertices of a face, we introduce some cumulative weights that are useful for the computation of the above *distsum*. For any f in the current block, the following formulas compute cumulative weights visiting f in clockwise and counterclockwise order, respectively.

$$\begin{array}{ll} \widehat{W}^R(n_f) = W_{v_f(n_f)} & \\ \widehat{W}^R(h, h+1) = \widehat{W}^R(h+1) + W_{v_f(h),v_f(h+1)} & h = n_f - 1, \dots, 1 \\ \widehat{W}^R(h) = \widehat{W}^R(h, h+1) + W_{v_f(h)} & h = n_f - 1, \dots, 1 \end{array} \tag{8}$$

and, similarly:

$$\begin{array}{ll} \widehat{W}^L(1) = W_{v_f(1)} & \\ \widehat{W}^L(h-1, h) = \widehat{W}^L(h-1) + W_{v_f(h-1),v_f(h)} & h = 2, \dots, n_f \\ \widehat{W}^L(h) = \widehat{W}^L(h-1, h) + W_{v_f(h)} & h = 2, \dots, n_f \end{array} \tag{9}$$

Suppose that $f = F_B$ and $D^L(v_f(h-1))$ has been computed for some $h \in \{2, \dots, n_f\}$. When the counterclockwise path from H to $v_f(h-1)$ is extended to $v_f(h)$, for some vertices of f their distance to the path decreases, while for other vertices it remains the same. In particular, if $n_f + h$ is even, the distance of the vertices having local label from h to $\frac{n_f+h}{2}$ decreases by 1; on the other hand, if $n_f + h$ is odd, the distance of the vertices having local label from h to $\lfloor \frac{n_f+h}{2} \rfloor$ decreases by 1. In addition, if $(v_f(h-1), v_f(h))$ is a chord of G, then, by Property 2, the path must contain $\Pi(v_f(h-1), v_f(h))$ and, therefore, the *distsum* of the enlarged path decreases by $Sav(v_f(h-1), v_f(h))$. A similar argument holds when $D^R(v_f(h+1))$ has been computed and the clockwise path from H to $v_f(h+1)$ is extended to $v_f(h)$. Hence, for $f = F_B$, we obtain:

$$D^L(v_f(h)) = \begin{cases} D(c_B) & \text{if } h = 2 \\ \quad -(\widehat{W}^R(h) - \widehat{W}^R(\frac{n_f+h}{2}, \frac{n_f+h}{2} + 1)) & \text{and } n_f + h \text{ is even} \\ \quad -Sav(v_f(h-1), v_f(h)) & \\ & \\ D(c_B) & \text{if } h = 2 \\ \quad -(\widehat{W}^R(h) - \widehat{W}^R(\lceil \frac{n_f+h}{2} \rceil)) & \text{and } n_f + h \text{ is odd} \\ \quad -Sav(v_f(h-1), v_f(h)) & \\ & \\ D^L(v_f(h-1)) & \text{if } h = 3, ..., n_f \\ \quad -(\widehat{W}^R(h) - \widehat{W}^R(\frac{n_f+h}{2}, \frac{n_f+h}{2} + 1)) & \text{and } n_f + h \text{ is even} \\ \quad -Sav(v_f(h-1), v_f(h)) & \\ & \\ D^L(v_f(h-1)) & \text{if } h = 3, ..., n_f \\ \quad -(\widehat{W}^R(h) - \widehat{W}^R(\lceil \frac{n_f+h}{2} \rceil)) & \text{and } n_f + h \text{ is odd} \\ \quad -Sav(v_f(h-1), v_f(h)) & \end{cases} \tag{10}$$

$$D^R(v_f(h)) = \begin{cases} D(c_B) & \text{if } h = n_f \\ \quad -(\widehat{W}^L(h) - \widehat{W}^L(\frac{h}{2}, \frac{h}{2}+1)) & \text{and } n_f \text{ is even} \\ \quad -Sav(v_f(h), v_f(1)) & \\ & \\ D(c_B) & \text{if } h = n_f \\ \quad -(\widehat{W}^L(h) - \widehat{W}^L(\lceil \frac{n_f}{2} \rceil)) & \text{and } n_f \text{ is odd} \\ \quad -Sav(v_f(h), v_f(1)) & \\ & \\ D^R(v_f(h+1)) & \text{if } h = n_f - 1, ..., 2 \\ \quad -(\widehat{W}^L(h) - \widehat{W}^L(\frac{h}{2}, \frac{h}{2}+1)) & \text{and } h \text{ is even} \\ \quad -Sav(v_f(h+1), v_f(h)) & \\ & \\ D^R(v_f(h+1)) & \text{if } h = n_f - 1, ..., 2 \\ \quad -(\widehat{W}^L(h) - \widehat{W}^L(\lceil \frac{h}{2} \rceil)) & \text{and } h \text{ is odd} \\ \quad -Sav(v_f(h+1), v_f(h)) & \end{cases} \tag{11}$$

with $D(c_B) = S_{c_B}$.

When $f \neq F_B$, similar formulas can be provided for computing the *distsum* of a counterclockwise path from H to $v_f(h)$ in f by updating the *distsum* of the previous counterclockwise path from H to $v_f(h-1)$ in f (the same holds for clockwise paths). However, in this case, four quantities must be computed, that is, $D^L_{r_f}(u)$, $D^L_{r_f,t_f}(u)$, $D^R_{t_f}(u)$ and $D^R_{r_f,t_f}(u)$. These formulas are given in the Appendix.

The following procedure to find a median path in an outerplanar graph is based on the previous formulas.

algorithm `MEDIANPATH`

input: A weighted outerplanar graph G and its representation tree $\mathcal{T}$.
output: The *distsum* D^* of a median path P^* of G.

```
begin
    $D^* = \infty$
    for each block and each leaf bridge $H$
        Root $\mathcal{T}$ at $H$ and let $\mathcal{T}_H$ be the resulting rooted tree
        PREPROCESSING($\mathcal{T}_H$)
        Compute the distsum $D_H$ of any hamiltonian path of $H$ by formula (4)
        if $D_H < D^*$ then $D^* = D_H$
        for each vertex $u$ in $H$ $D_H(u) = D_H$
        Visit $\mathcal{T}_H$ in BFS order
        for each visited $B \neq H$ of $\mathcal{T}_H$ do
            Let $c_B$ be the source of $B$
            if $B$ is a bridge $(c_B, u)$ then
                Compute $D_H(u)$ by formula (5)
            else ($B$ is a block)
                Number the faces of $B$ from 1 to $F_B$ such that $c_B \in F_B$ and each face $f$ is
                adjacent to exactly one face $f' > f$
            for $f = F_B, ..., 1$
                MEDIANPATHFACE($f$)
            end for
        end for
    end for
end
```

```
algorithm MEDIANPATHFACE($f$)

input: A face $f$, weights $W_u$, $\forall u \in V(f)$, $W_{uv}$ and $Sav(u,v)$, $\forall (u,v) \in E(f)$.
output: $\forall u \in V(f)$ the distsum $D_H(u)$ of a best path from $H$ to $u$ and the distsum
$D^*$ of the best current path starting from $H$.
begin
    Let $n_f$ be the number of vertices of $f$
    Starting from $c_B$ (if $f = F_B$) or from $r_f$ (if $f \neq F_B$)
    assign counterclockwise local labels $1, ..., n_f$ to the vertices of $f$
    for $h = n_f, ..., 1$
        Compute $\widehat{W}^R(h)$
        if $h \neq n_f$ then compute $\widehat{W}^R(h, h+1)$
    end for
    for $h \in 1, ..., n_f$
        Compute $\widehat{W}^L(h)$
        if $h \neq 1$ then compute $\widehat{W}^L(h-1, h)$
    end for
    if $f = F_B$ then
        for all $v_f(h)$ in $f$ such that $v_f(h) \neq c_B$
            Compute $D^L(v_f(h))$ and $D^R(v_f(h))$
            $D(v_f(h)) = \min\{D^L(v_f(h)), D^R(v_f(h))\}$
            if $D(v_f(h)) < D^*$ then $D^* = D(v_f(h))$
        end for
    else ($f \neq F_B$)
        for all $v_f(h)$ in $f$ such that $v_f(h) \neq r_f, t_f$
            Compute $D^L_{r_f}(v_f(h))$ and $D^L_{r_f,t_f}(v_f(h))$
            Compute $D^R_{t_f}(v_f(h))$ and $D^R_{r_f,t_f}(v_f(h))$
            $D^L(v_f(h)) = \min\{D^L_{r_f}(v_f(h)), D^L_{r_f,t_f}(v_f(h))\}$
            $D^R(v_f(h)) = \min\{D^R_{t_f}(v_f(h)), D^R_{r_f,t_f}(v_f(h))\}$
            $D(v_f(h)) = \min\{D^L(v_f(h)), D^R(v_f(h))\}$
            if $D(v_f(h)) < D^*$ then $D^* = D(v_f(h))$
        end for
end
```

Proposition 3 *Algorithm MEDIANPATH correctly finds the* distsum *of a median path on an outerplanar graph G in $O(n^2)$ time.*

Proof. On the basis of the properties listed in Section 2, an optimal path P^* in G that ends in two blocks or bridges of G, say B_1 and B_2, contains all the vertices of B_1 and B_2. Taking into account the structure of an outerplanar graph, and exploiting the possibility of representing it through the representation tree $\mathcal{T}$, our algorithm enumerates all the possible paths that can be generated in G with such properties and, among them, selects the one with the minimum *distsum.* For the time complexity result, we first observe that in our algorithm the representation tree is rooted at each block or bridge, hence, since G can be decomposed into k blocks and bridges, the graph is visited at most $O(k)$ times. For each rooted representation tree the preprocessing phase requires $O(n)$ time (see Proposition 2). On the other hand, the algorithm computes the *distsum* of any hamiltonian path of H in constant time, while the *distsum* $D_H(u)$ for all the $u \in V(H)$ are initialized in $O(|V(H)|)$ time. In the rest of the procedure, for each visited block B, the analysis requires a time linear in the number of vertices of B. In fact, if B is a bridge it requires a constant time. On the other hand, if B is a biconnected block, the procedure MEDIANPATHFACE(f) is performed for each face f of B in $O(n_f)$ time. This implies that the analysis of each block requires $O(|V(B)|)$ time. In conclusion, for each rooted representation tree $\mathcal{T}_H$, our algorithm requires $O(n)$ time. Hence, the overall time complexity of the algorithm is $O(kn)$. Notice that, since $k = O(n)$, the time complexity of our algorithm could be $O(n^2)$. $\square$

5 Conclusion

The algorithm proposed in this paper solves the Median Path Problem on an outerplanar graph G in $O(kn)$ time, where k is the number of blocks and bridges of G and n is the number of its vertices. The problem is solved for the case in which nonnegative weights are associated to the vertices of G, while equal weights are assigned to the edges. Notice that our algorithm cannot be generalized in a straightforward way to the more general case in which arbitrary positive weights are associated also to the edges. We shall leave the study of this problem to future work, but, in the following, we provide a result that could help in the analysis of the computational complexity of the problem, giving some intuition with it.

PROBLEM 1. Let $G = (V, E)$ be an outerplanar graph; $w : V \to \mathbb{R}^+$ a weighting function that assigns a weight w_v to each $v \in V$; $\ell : E \to \mathbb{R}^+$ a weighting function that assigns a weight $\ell(e)$ to each $e \in E$. Finally, let $K > 0$.
Find a path P in G such that the sum of the weighted distances to P from the vertices not in P is exactly equal to K, that is, the *distsum* of P is K.

Proposition 4 *PROBLEM 1. is NP-complete:*

Proof. The proof is by reduction from the following problem that is shown to be NP-Complete in [12].

PROBLEM 2. Two vectors of integers, $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$, are given such that

$$\sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i = T.$$

Find a subset of indices $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} a_i + \sum_{i \notin S} b_i = \frac{T}{2}$.

We show that PROBLEM 2 can be reduced to PROBLEM 1.

Consider the weighted outerplanar graph G shown in Figure 5, where all the vertices with degree equal to 2 have weight equal to 1, while the remaining vertices have a weight equal to a quantity $Q >> T$. Except for the two vertices A and B, and the edges (A, A') and (B', B) (with weight equal to 1) the graph consists of n copies of a cycle (with 4 vertices) that share pairwise a common vertex with degree 4. We number these cycles from 1 to n: in cycle i we denote by A_i and B_i the two vertices of degree 2 with weight equal to 1, respectively, and assign a weight equal to a_i to the two edges incident to A_i and a weight equal to b_i to the two edges incident to B_i (see Figure 5).

Set $K = \frac{T}{2}$ and consider a solution of PROBLEM 2 and the corresponding subset S. Then, it is always possible to build in G a path P from A to B with *distsum* equal to K by including in P edges (A, A'), (B', B) and the pair of edges incident to vertex A_i if $i \notin S$, or the pair of edges incident to vertex B_i if $i \in S$. In fact, for such P we have:

$$\sum_{v \notin P} d(v,P) = \sum_{i \notin S} b_i + \sum_{i \in S} a_i = \frac{T}{2}.$$

On the other hand, suppose that P is a path in G with *distsum* equal to $K = \frac{T}{2}$. Let S be the subset of indices of the vertices A_i in G that does not belong to P. Then we have:

$$\frac{T}{2} = \sum_{v \notin P} d(v,P) = \sum_{i \notin S} b_i + \sum_{i \in S} a_i.$$

□

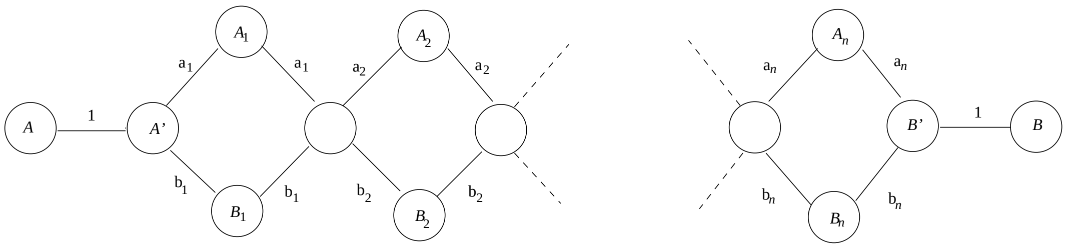

Figure 5: The outerplanar graph used for the reduction from PROBLEM 2.

To conclude this section, we notice that the algorithm presented in this paper solves MPP on general outerplanar graphs: the only assumption is that it is connected. We already noticed that when the outerplanar graph G is biconnected, the solution of MPP is trivial, but this is not true if we consider the case of finding a median path P between two fixed end vertices. Once a pair of vertices s and t are fixed, one may be interested in finding a path in G that connects s and t and minimizes the sum of the weighted distances from all the vertices of G to P. Our algorithm can be applied to this problem directly, since in this case the graph G consists of a single block: starting from s, we apply the same formulas provided in Section 4 and the optimal path from s to t is computed in time linear in the number of vertices of the block.

APPENDIX

$$D^L_{r_f}(v_f(h)) = \begin{cases} D^L(r_f) & \text{if } h=1 \\ \\ D^L_{r_f}(v_f(h-1)) & \text{if } h=2,...,n_f-1 \\ \quad -(\widehat{W}^R(h)-\widehat{W}^R(\frac{n_f+h}{2},\frac{n_f+h}{2}+1)) & \text{and } n_f+h \text{ is even} \\ \quad -Sav(v_f(h-1),v_f(h)) & \\ \\ D^L_{r_f}(v_f(h-1)) & \text{if } h=2,...,n_f-1 \\ \quad -(\widehat{W}^R(h)-\widehat{W}^R(\lceil\frac{n_f+h}{2}\rceil)) & \text{and } n_f+h \text{ is odd} \\ \quad -Sav(v_f(h-1),v_f(h)) & \end{cases} \tag{12}$$

$$D^L_{r_f,t_f}(v_f(h)) = \begin{cases} D^R(r_f)+Sav(r_f,t_f) & \text{if } h=1 \\ \\ D^L_{r_f,t_f}(v_f(h-1)) & \text{if } h=2,...,n_f-1 \\ \quad -(\widehat{W}^R(h)-\widehat{W}^R(\frac{n_f+h}{2})) & \text{and } n_f+h \text{ is even} \\ \quad -Sav(v_f(h-1),v_f(h)) & \\ \\ D^L_{r_f,t_f}(v_f(h-1)) & \text{if } h=2,...,n_f-1 \\ \quad -(\widehat{W}^R(h)-\widehat{W}^R(\lfloor\frac{n_f+h}{2}\rfloor,\lceil\frac{n_f+h}{2}\rceil)) & \text{and } n_f+h \text{ is odd} \\ \quad -Sav(v_f(h-1),v_f(h)) & \end{cases} \tag{13}$$

$$D^R_{t_f}(v_f(h)) \begin{cases} D^R(t_f) & \text{if } h=n_f \\ \\ D^R_{t_f}(v_f(h+1)) & \text{if } h=n_f-1,\dots,2 \\ \quad -(\widehat{W}^L(h)-\widehat{W}^L(\frac{h}{2})) & \text{and } h \text{ is even} \\ \quad -Sav(v_f(h),v_f(h+1)) & \\ \\ D^R_{t_f}(v_f(h+1) & \text{if } h=n_f-1,\dots,2 \\ \quad -(\widehat{W}^L(h)-\widehat{W}^L(\lfloor\frac{h}{2}\rfloor,\lceil\frac{h}{2}\rceil)) & \text{and } h \text{ is odd} \\ \quad -Sav(v_f(h),v_f(h+1)) & \end{cases} \tag{14}$$

$$D^R_{r_f,t_f}(v_f(h)) = \begin{cases} D^L(t_f)+Sav(r_f,t_f) & \text{if } h=n_f \\ \\ D^R_{r_f,t_f}(v_f(h+1)) & \text{if } h=n_f-1,\dots,2 \\ \quad -(\widehat{W}^L(h)-\widehat{W}^L(\frac{h}{2},\frac{h}{2}+1)) & \text{and } h \text{ is even} \\ \quad -Sav(v_f(h),v_f(h+1)) & \\ \\ D^R_{r_f,t_f}(v_f(h+1)) & \text{if } h=n_f-1,\dots,2 \\ \quad -(\widehat{W}^L(h)-\widehat{W}^L(\lceil\frac{h}{2}\rceil)) & \text{and } h \text{ is odd} \\ \quad -Sav(v_f(h),v_f(h+1)) & \end{cases} \tag{15}$$

References

[1] Alstrup S., Lauridsen P.W., Sommerlund P., and Thorup M., Finding cores of limited length, Techincal Report, The IT University of Copenhagen, 2001.

[2] Avella P., Boccia M., Sforza A., Vasil'ev I., A branch-and-cut algorithm for the median-path problem, *Computational Optimization and Applications*, 32 (2005), 215-230.

[3] Becker R.I., Lari I., Scozzari A., Storchi G., The location of median paths on grid graphs, *Annals of Operations Research*, 150 (2007), 65-78.

[4] Becker R.I., Chiang Y.I., Lari I., Scozzari A., Storchi G., Finding the ℓ-core of a tree, *Discrete Applied Mathematics*, 118 (2002), 25-42.

[5] Hakimi S.L., Schmeichel E.F., Labbé M., On locating path- or tree- shaped facilities on networks, *Networks*, 23 (1993), 543-555.

[6] Harary F., Graph Theory. Reading, MA: Addison-Wesley, 1994.

[7] Lari I., Ricca F., Scozzari A., Comparing different metaheuristic approaches for the median path problem with bounded length, *European Journal of Operational Research*, to appear (doi:10.1016/j.ejor.2007.07.001).

[8] Lari I., Ricca F., Scozzari A., The forest wrapping problem on outerplanar graphs, *Lecture Notes in Computer Science*, 2573 (2002), 345-354.

[9] Minieka E., The optimal location of a path or tree in a tree network, *Networks*, 15 (1985), 309-321.

[10] Morgan C.A., Slater J.P., A linear Algorithm for a core of a tree, *Journal of Algorithms*, 1 (1980), 247-258.

[11] Peng S., Stephens A.B., Yesha Y., Algorithms for a core and a k-tree core of a tree, *Journal of Algorithms*, 15 (1993), 143-159.

[12] Richey M.B., Optimal location of a path or tree on a network with cycles, *Networks*, 20 (1990), 391-407.

[13] Slater P.J., Locating central paths in a graph, *Transportation Science*, 16 (1982), 1-18.