Locating median paths on connected outerplanar graphs

Andrea Scozzari
University of Rome "La Sapienza"

andrea.scozzari@uniroma1.it

co-authors:
Isabella Lari
Federica Ricca
Ronald Becker

7th Cologne-Twente Workshop on Graphs and Combinatorial Optimization
Università degli Studi di Milano
Gargnano, Italy, May 13-15, 2008

Median-path in the Literature

Path-Median Problem is polynomial on trees:

Without restrictions on the length of the path

Morgan and Slater, J. of Alg. (1980)
Becker, Quaest. Math. (1990)
Peng, Stephens and Yesha, J. of Alg. (1993)

With restrictions on the length of the path

Minieka, Networks (1985)
Peng and Lo, J. of Alg. (1996)
Becker, Chang, Lari, **Scozzari** and Storchi, DAM (2002)
Alstrup, Lauridsen, Sommerlund and Thorup, Tech. Rep. (2001)

Path-Median Problem with restricted length is NP-complete on networks:

Hakimi, Schmeichel and Labbé, Networks (1993)	[planar g. with vertex deg. ≤ 5]
Richey, Networks (1990)	[series-parallel graphs]
Becker, Lari, **Scozzari**, Storchi, Annals of Operations Research (2007)	[grid graphs]

The path-median problem without restrictions on the length of the path has not been studied yet on networks with cycles.

The median path problem in connected outerplanar graphs

Given

- a connected outerplanar graph $G = (V,E)$, $|V| = n$, $|E| = m$
- weights equal to 1 assigned to each edge
- weights $w(v) \geq 0$ associated with each vertex v
- distances $d(u,v)$ = shortest path from u to v on G, for each pair of vertices u, v
- distances $d(v,P)$ from v to the closest vertex in P

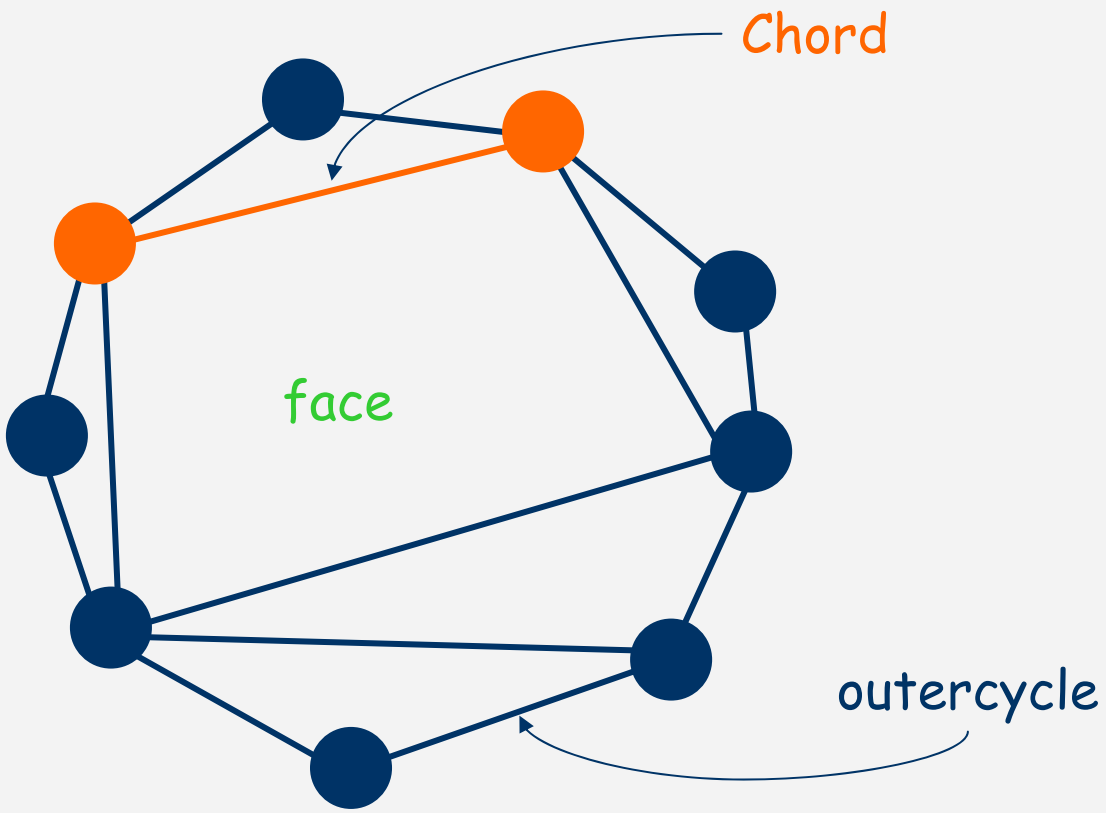
Problem:

find in G a path P^* (of any length) which minimizes the sum of the weighted distances from each vertex in G to P^* (the *distsum* of P^*):

$$W(P^*) = \min_{P \in \Pi} \sum_v w(v)\, d(v,P)$$

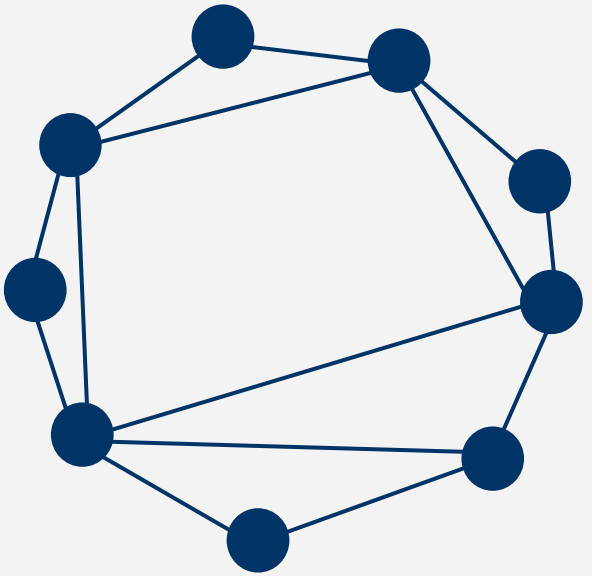
Outerplanar graphs

An outerplanar graph is a planar graph that can be embedded in the plane in such a way that all the vertices lie on the boundary (outercycle).



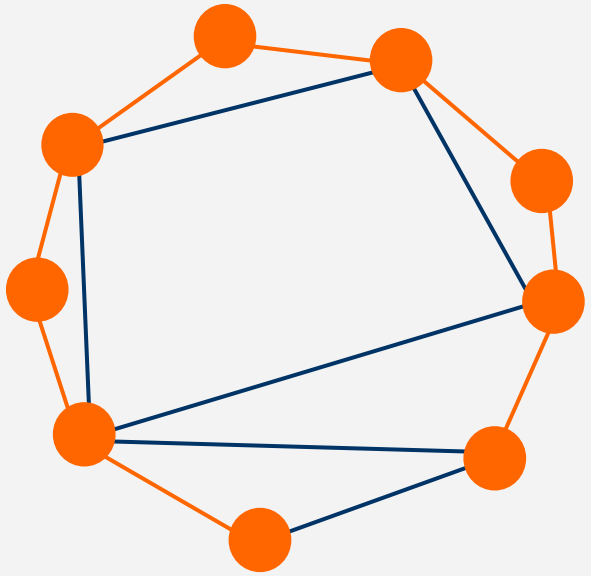
The median path problem in connected outerplanar graphs

Biconnected
outerplanar graph



The median path problem in connected outerplanar graphs

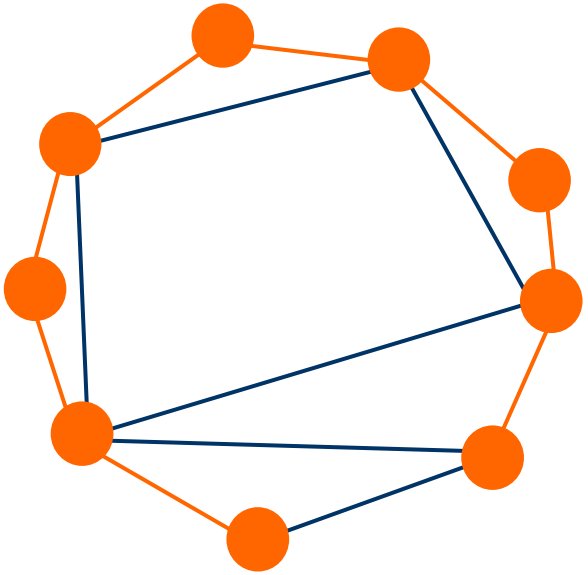
Biconnected outerplanar graph



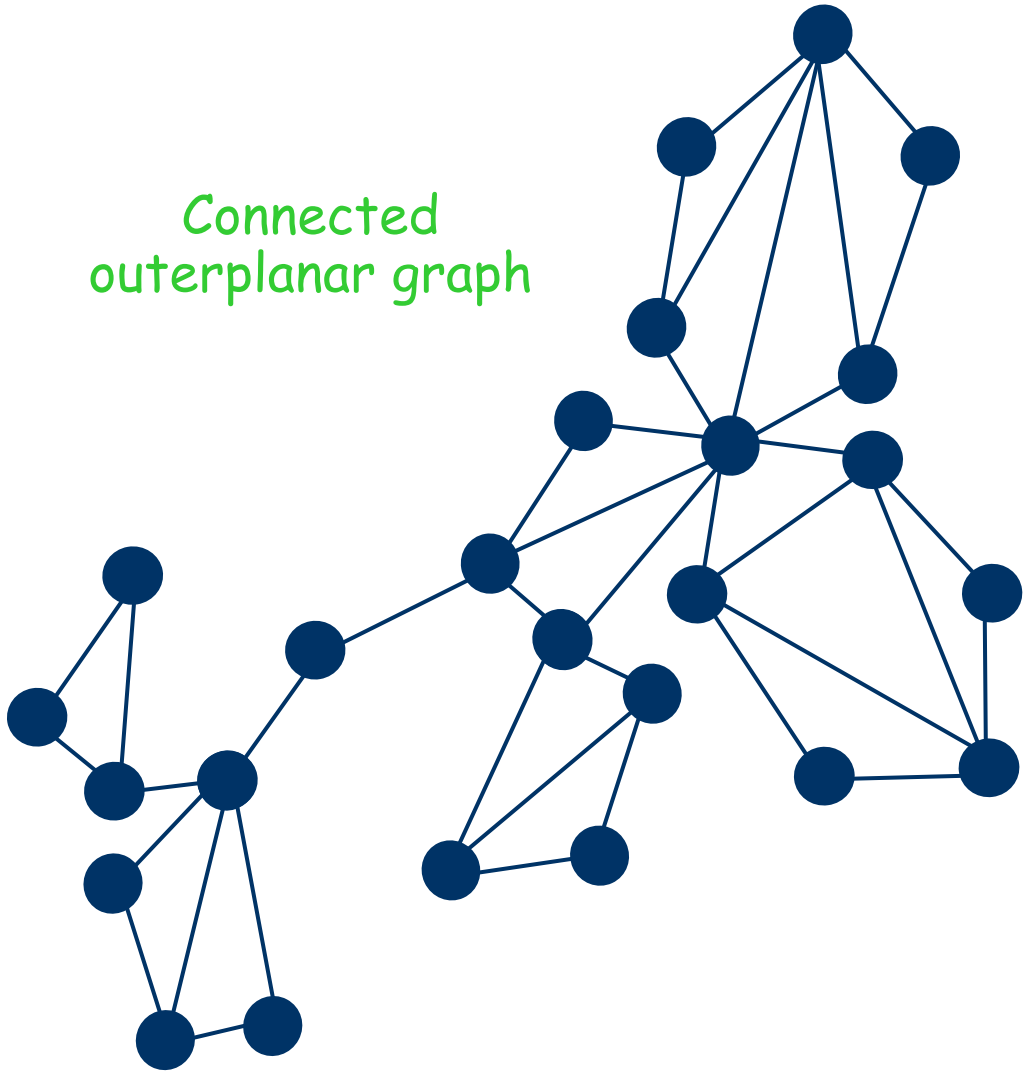
Note: A hamiltonian path always exists

The median path problem in connected outerplanar graphs

Biconnected outerplanar graph

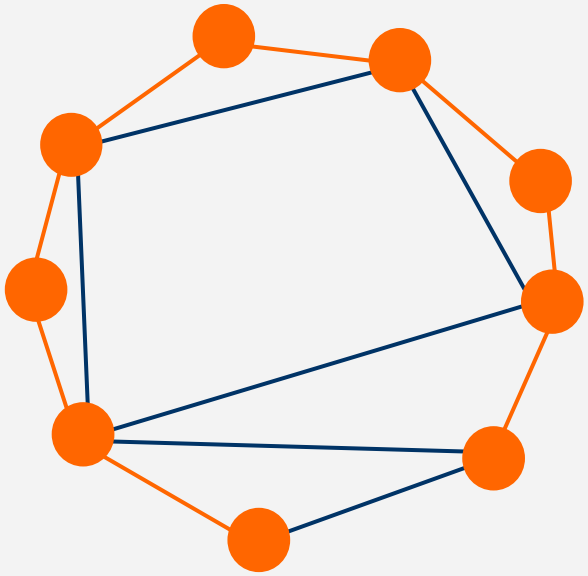


Connected outerplanar graph

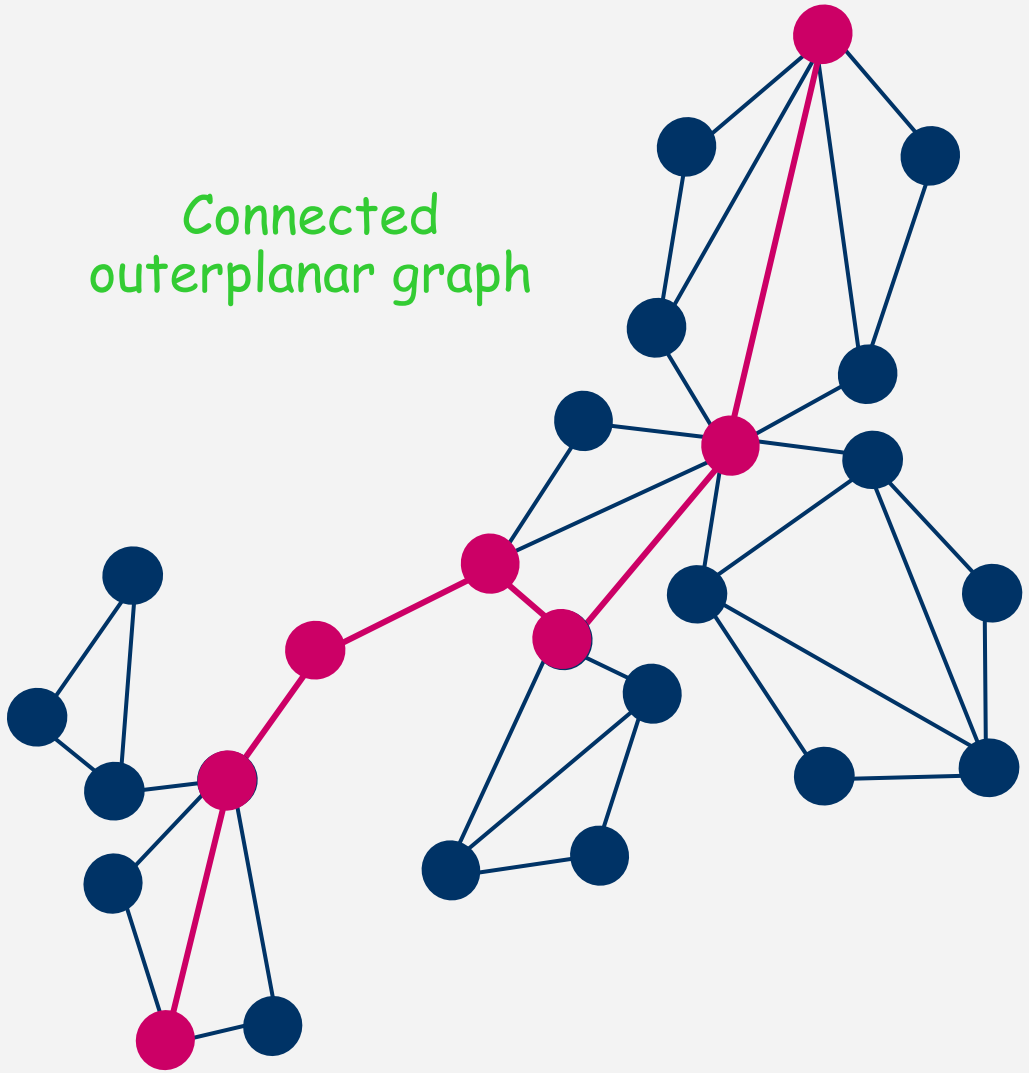


The median path problem in connected outerplanar graphs

Biconnected outerplanar graph

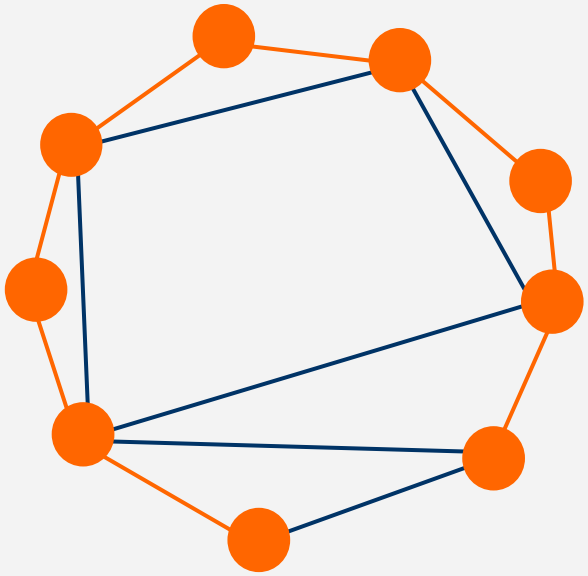


Connected outerplanar graph

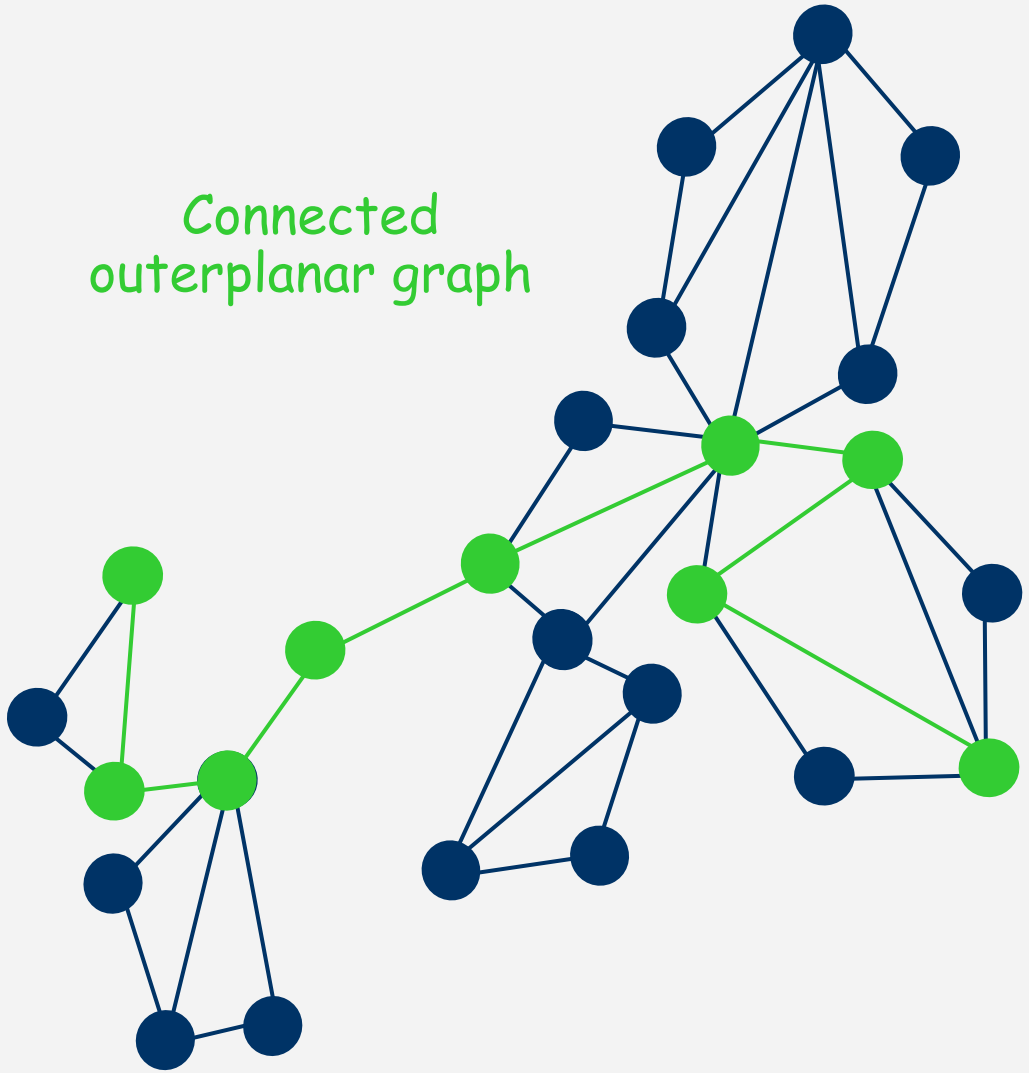


The median path problem in connected outerplanar graphs

Biconnected outerplanar graph



Connected outerplanar graph

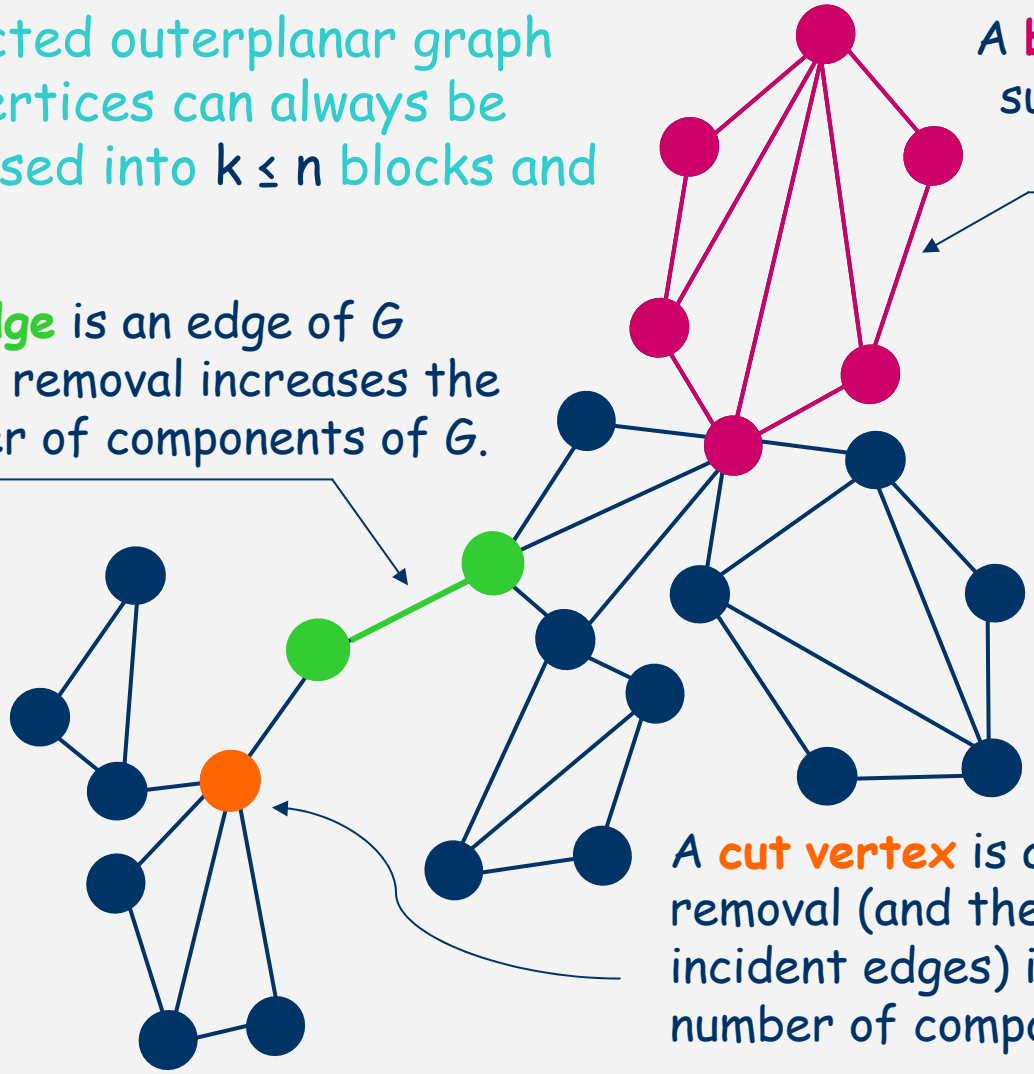


Decomposition into blocks and bridges

A connected outerplanar graph with n vertices can always be decomposed into $k \leq n$ blocks and bridges.

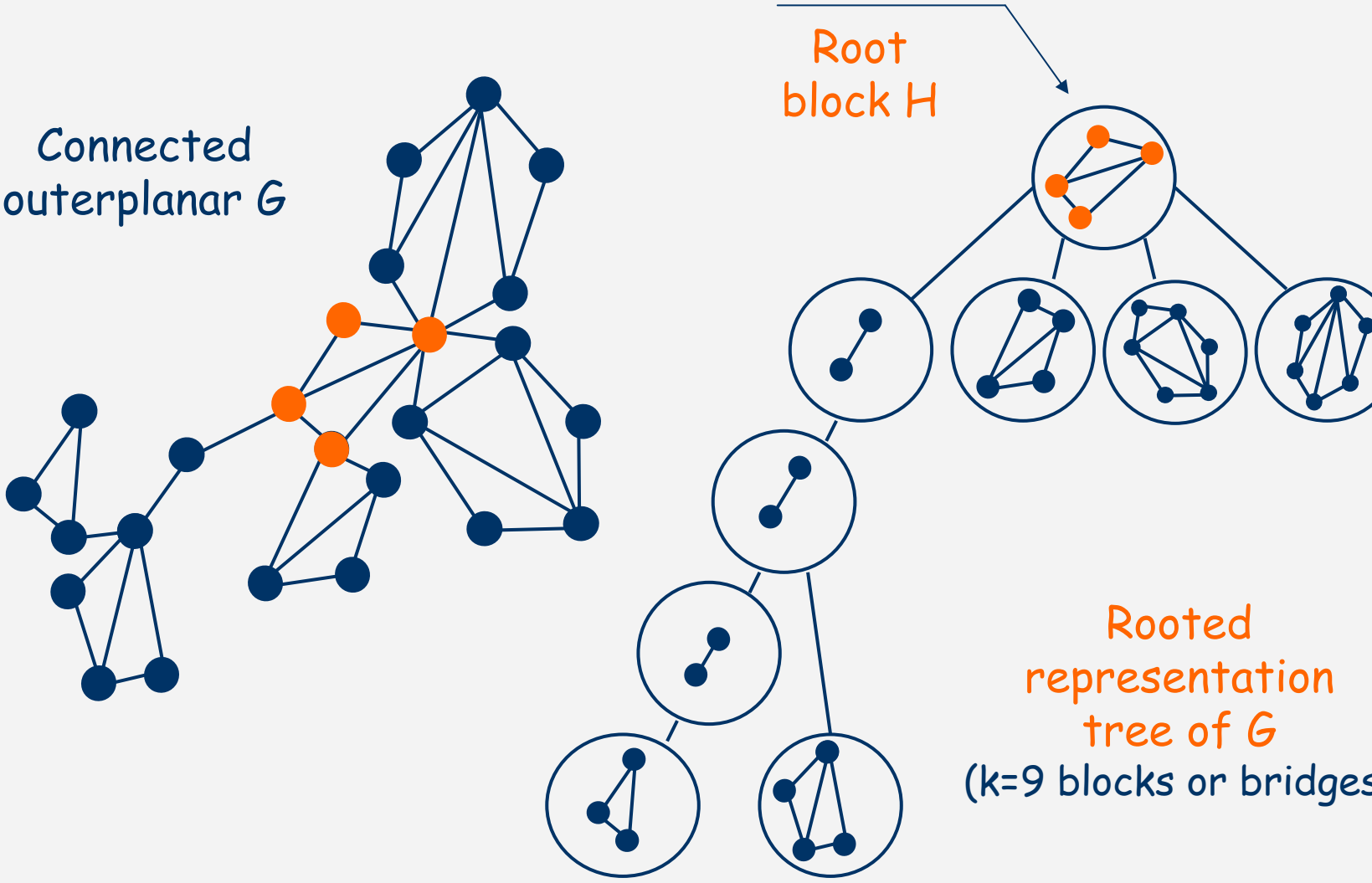
A **block** is a maximal subgraph of G with no cut vertices.

A **bridge** is an edge of G whose removal increases the number of components of G.



A **cut vertex** is a vertex whose removal (and the removal of all its incident edges) increases the number of components of G.

Representation tree



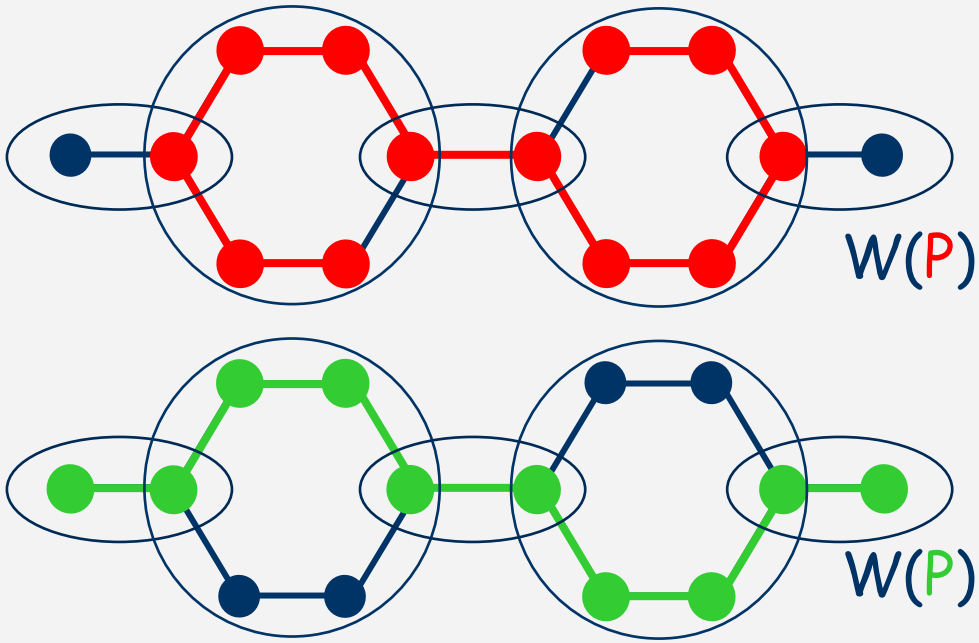
Properties

Let P* be an optimal solution to the median path problem on G.

Let $\mathcal{P}^*$ be the path in T given by the set of blocks and bridges traversed by P*.

Property 1.
The end vertices of $\mathcal{P}^*$ are not necessarily leaves of T.

EXAMPLE
Suppose each vertex has weight equal to 1.



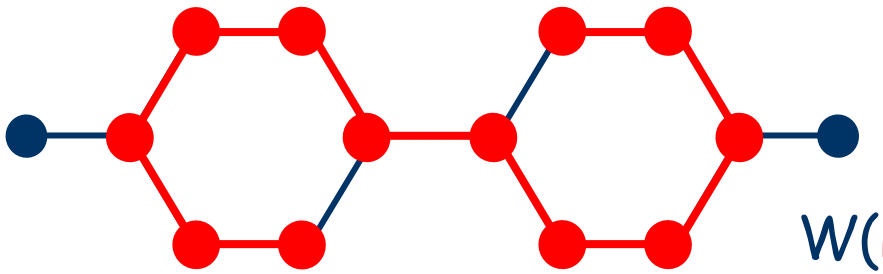
Properties

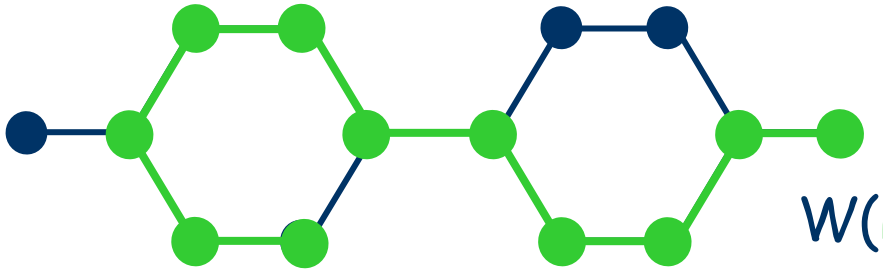
Let P* be an optimal solution to the median path problem on G.

Let $\mathcal{P}^*$ be the path in T given by the set of blocks and bridges traversed by P*.

Property 1.
The end vertices of $\mathcal{P}^*$ are not necessarily leaves of T.

EXAMPLE
Suppose each vertex has weight equal to 1.

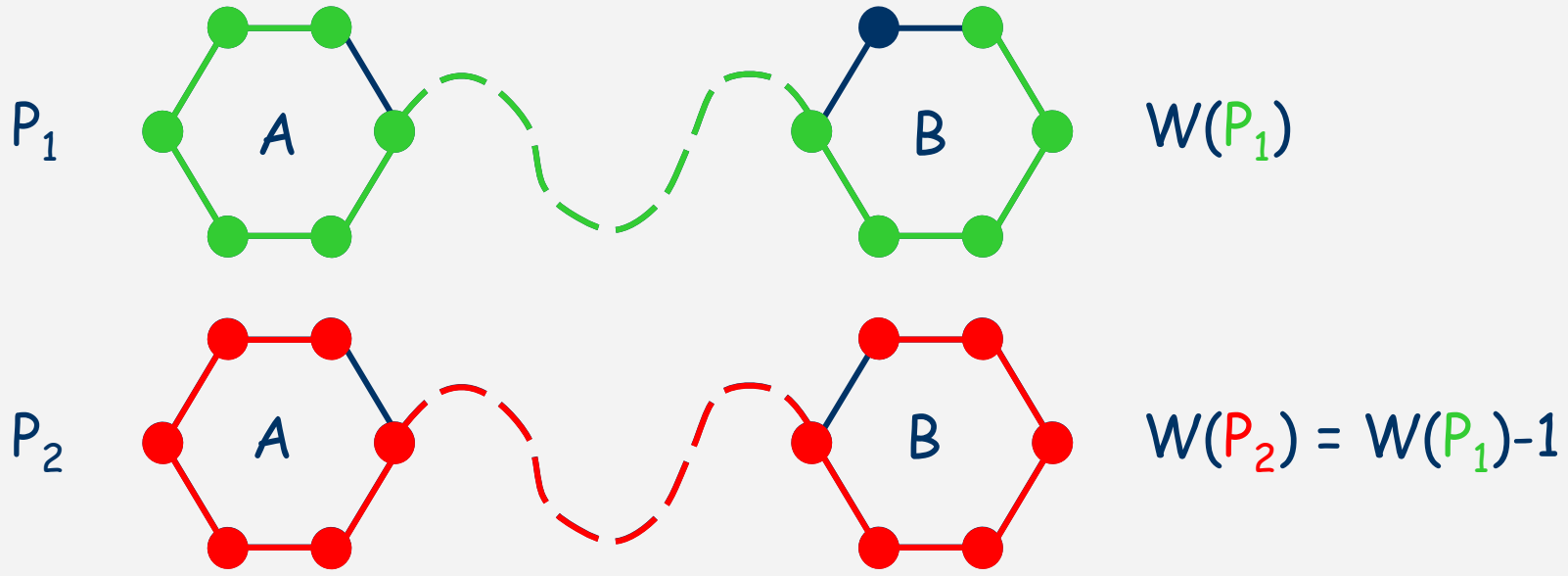




Properties

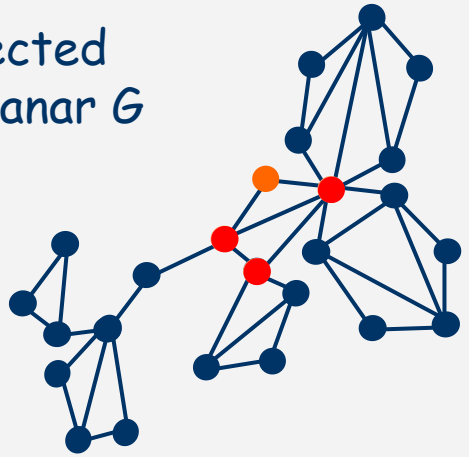
Property 2.
Every optimal path P* has a double-hook shape, that is, all the vertices included in the end blocks of $\mathcal{P}$* belong to P*.

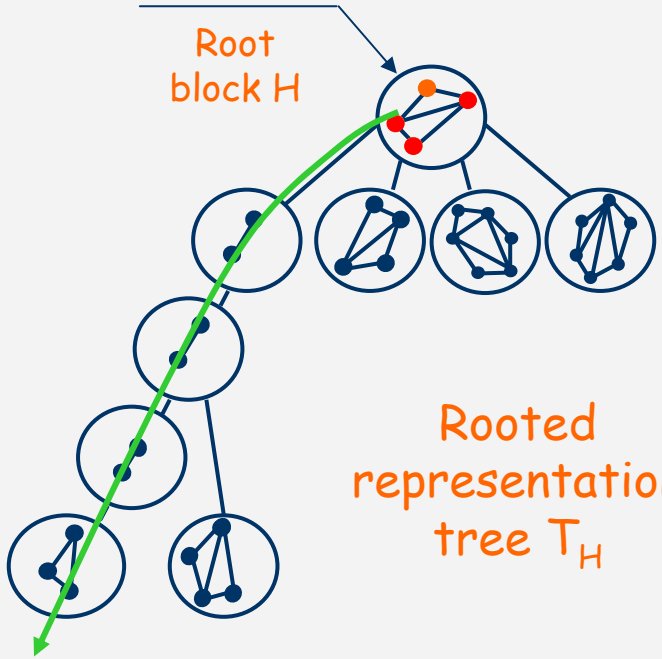
EXAMPLE
Suppose each vertex has weight equal to 1.
Let A and B be the two end blocks of $\mathcal{P}$*.



Idea of the algorithm

Connected outerplanar G





Rooted representation tree T_H

Property 2 ⇨ we root T at a block or bridge and search for optimal paths among those paths having one end block in the root of T.

Property 1 ⇨ we must root T at each possible block or bridge.

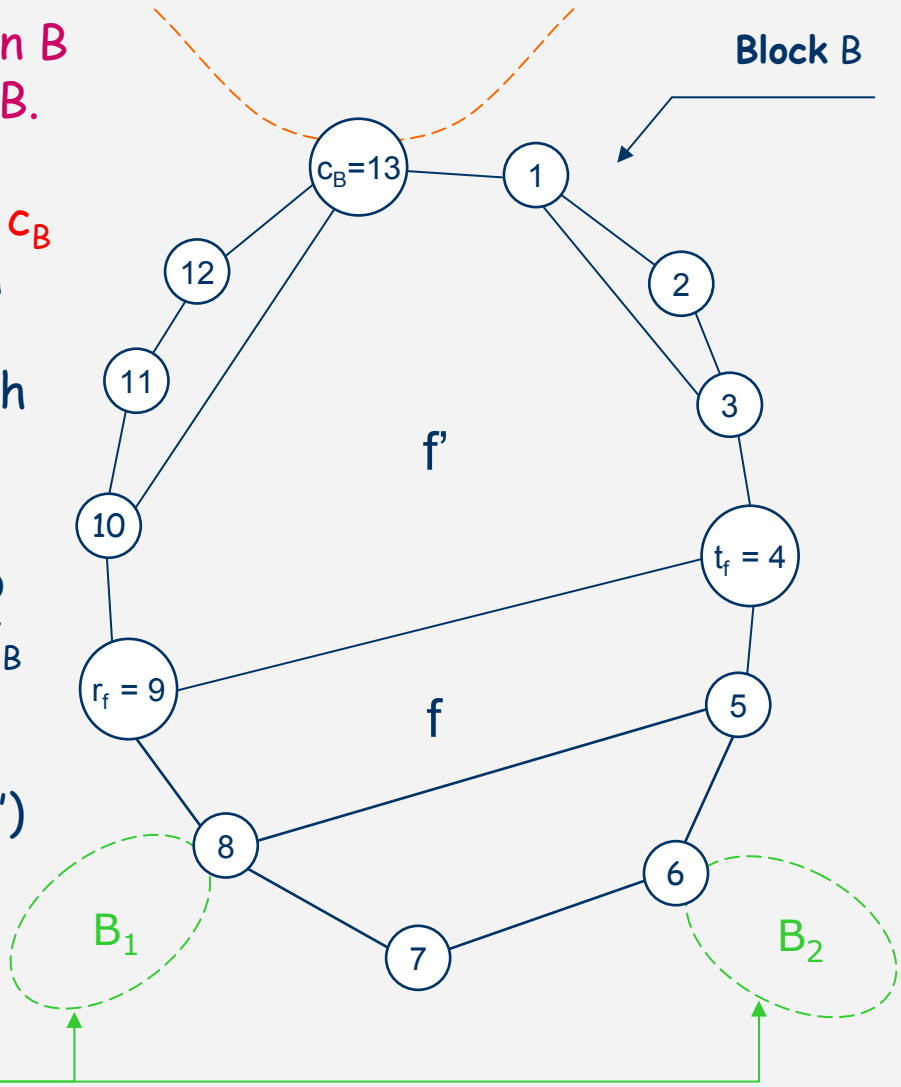
Some notation

For a given representation tree T_H rooted at block H

Let F_B be the number of faces in B and V(B) the set of vertices of B.

In each block $B \neq H$ we denote by c_B the cut vertex that B shares with its parent in T_H. We number the vertices of B from 1 to |V(B)| such that c_B has the largest number.

We assign a number $f = 1, \ldots, F_B$ to the faces of B such that c_B is in F_B and each face f is adjacent to exactly one face $f' > f$ (f has the chord (r_f, t_f) in common with f')



B_1 and B_2 are children of B in T_H and in T_B

Preprocessing

For a given representation tree T_H rooted at block H

In a preprocessing phase, in each block B of T_H, we compute the following quantities associated to the vertices of G:

- S_{c_B} the sum of the weighted distances to c_B from all the vertices of G belonging to the blocks in T_B
- W_{c_B} the sum of the weights of the vertices of G belonging to the blocks in T_B
- S_{r_f} the sum of the weighted distances to r_f from all the vertices of G belonging to $V(T_B)\setminus[V(B)\setminus V(r_f,t_f)]$ that are closer to r_f than to t_f (vertices *assigned* to r_f in f)
- W_{r_f} the sum of the weights of the vertices of G belonging to $V(T_B)\setminus[V(B)\setminus V(r_f,t_f)]$ that are *assigned* to r_f in f

S_{t_f} and W_{t_f} are defined similarly.

Actually, we compute the quantities W_u and S_u *for all* the vertices $u \in V$

The algorithm

For a given representation tree T_H rooted at block H

Once all the quantities have been computed in the preprocessing, the algorithm is performed through a visit of T_H in a BFS order.

At block H the algorithm computes the *distsum* of an hamiltonian path in H.

At each block $B \neq H$ the algorithm visits B face by face from $f=F_B$ to f=1. In each face f, and for each vertex v in f:

it computes the *distsum* of a best path from H to v in f, $D_H(v,f)$

At the end of the visit of T_H the algorithm records the best *distsum* found so far:

$$D_H = \min_{v \text{ in } f} D_H(v,f)$$

The algorithm applied to T_H runs in O(n) time.

Formulas

For a given representation tree T_H rooted at block H

The complexity result is due to the fact that we exploit the structure of our outerplanar graph G in order to arrange an efficient computation of the *distsum* $D_H(v,f)$, for all v in each face f.

At the beginning we compute *distsum* of any hamiltonian path in H, and initialize D_H at this value.

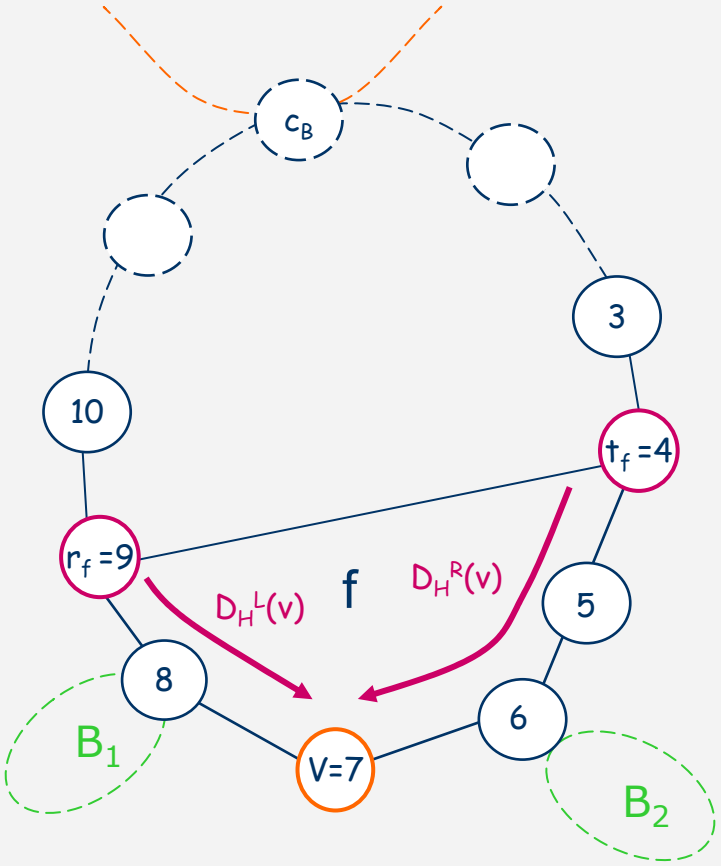
Going down in the BFS visit of the blocks $B \neq H$ in T_H, in each visited face of a block B the *distsum* of a best path from H to v in f, $D_H(v,f)$, is computed by updating the *distsum* of the best path from H to $r_{f'}$ or $t_{f'}$ of the unique face f'>f in B.

The updating is performed basically by computing the saving in the *distsum* obtained by attaching to the current path a new edge

Formulas

For a given representation tree T_H rooted at block H

The complexity result is due to the fact that we exploit the structure of our outerplanar graph G in order to arrange an efficient computation of the *distsum* $D_H(u)$, for all u in each face f.



Actually, since every vertex v lies in a face of G, when the algorithm visits face f, for each v in f, it computes the following two quantities:

$D_H^L(v)$ the minimum *distsum* of a path from H to u that visits f in a counterclockwise order (counterclockwise path);

$D_H^R(v)$ the minimum *distsum* of a path from H to u that visits f in a clockwise order (clockwise path).

The algorithm

By repeating both the preprocessing and the algorithm for each possible representation tree T_H, we are able to compute the *distsum* of an optimal median path P* as follows:

$$D^* = \min_{H \text{ in } T} D_H$$

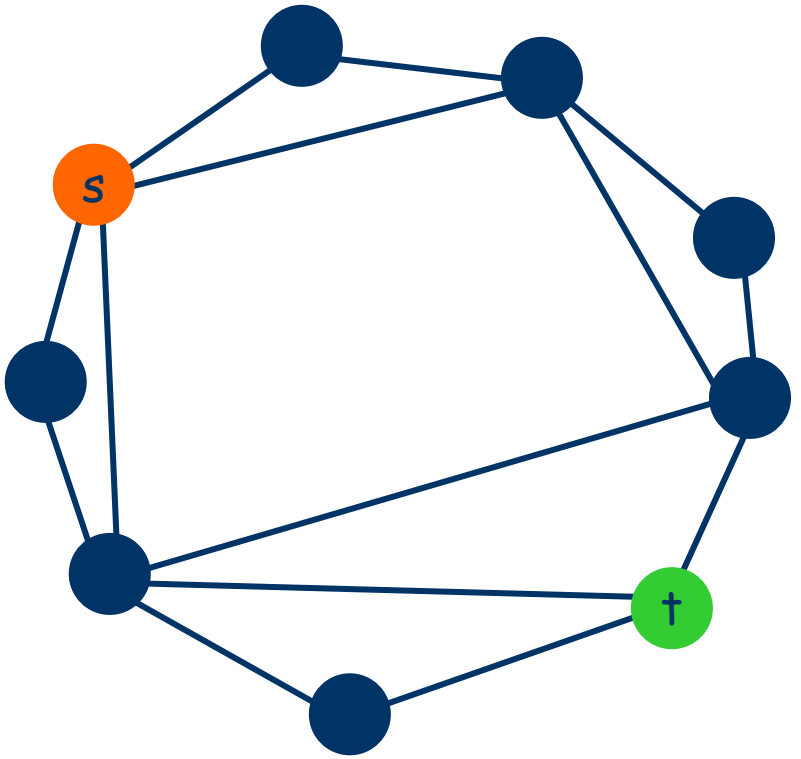
The overall time complexity (preprocessing + algorithm for all the possible T_H) runs is O(kn), where k is the number of blocks and bridges in which G can be decomposed.

The median path problem in biconnected outerplanar graphs with fixed end vertices

Suppose the two vertices s and t are fixed. Finding the median path with end vertices in s and t is not trivial even on biconnected outerplanar graphs.

EXAMPLE

Suppose each vertex has weight equal to 1.



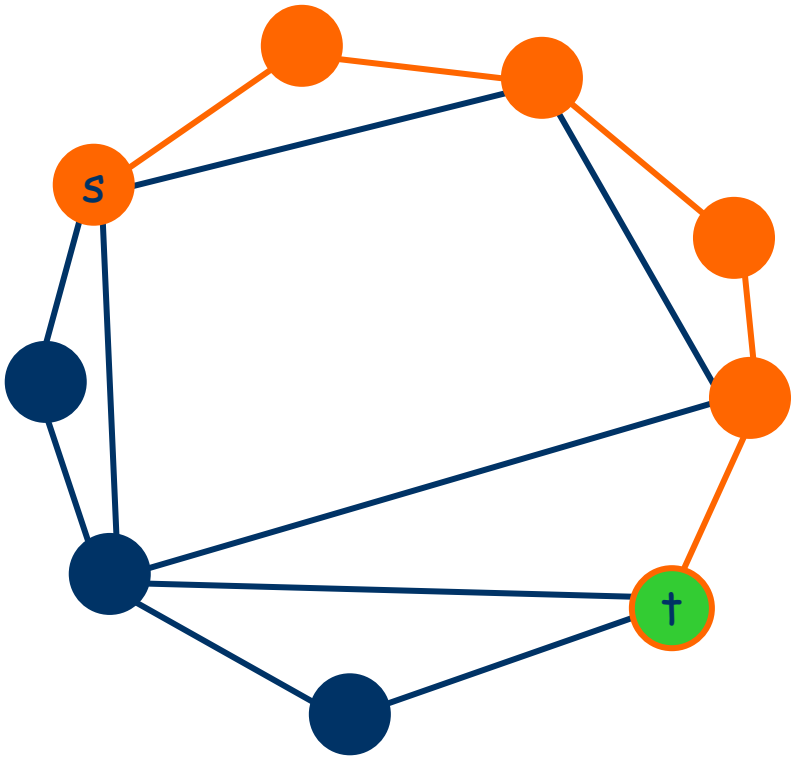
The median path problem in biconnected outerplanar graphs with fixed end vertices

Suppose the two vertices s and t are fixed. Finding the median path with end vertices in s and t is not trivial even on biconnected outerplanar graphs.

EXAMPLE

Suppose each vertex has weight equal to 1.

W(P) = 3



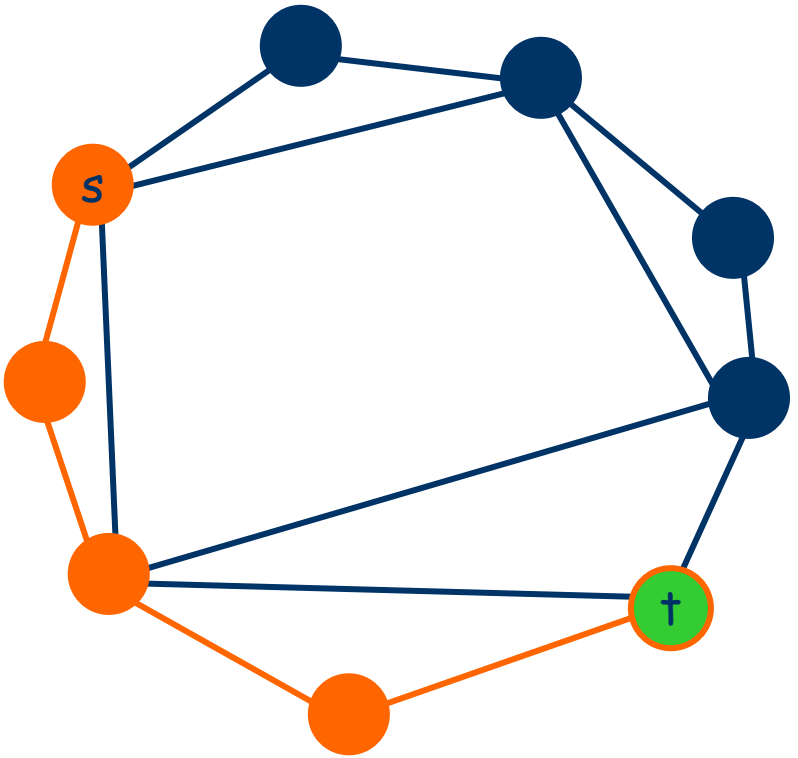
The median path problem in biconnected outerplanar graphs with fixed end vertices

Suppose the two vertices s and t are fixed. Finding the median path with end vertices in s and t is not trivial even on biconnected outerplanar graphs.

EXAMPLE

Suppose each vertex has weight equal to 1.

W(P) = 5

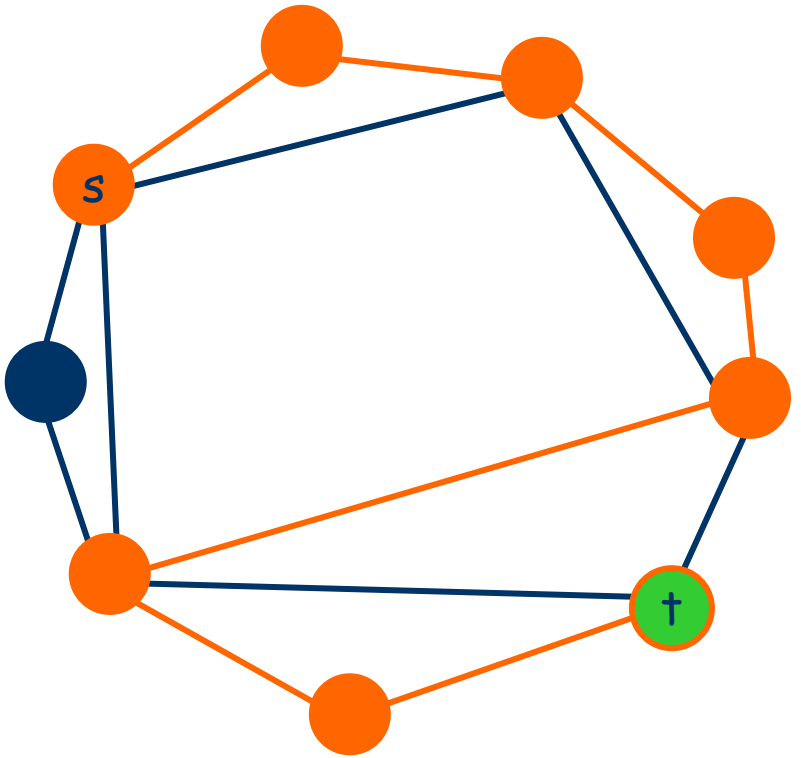


The median path problem in biconnected outerplanar graphs with fixed end vertices

Suppose the two vertices s and t are fixed. Finding the median path with end vertices in s and t is not trivial even on biconnected outerplanar graphs.

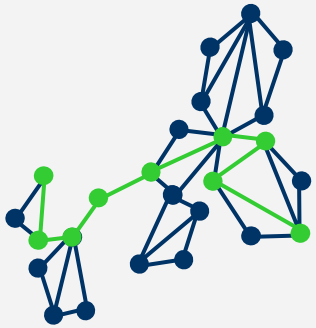
EXAMPLE

Suppose each vertex has weight equal to 1.



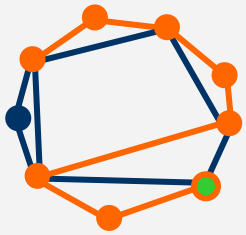
W(P) = 1

CONCLUSION



We provided a $O(kn)$ time algorithm for the median path problem on a connected outerplanar graph with n vertices and k blocks or bridges.

As a by-product, we also provided a linear time algorithm for the median path problem on a biconnected outerplanar graph for the case in which the ends of the path are fixed.



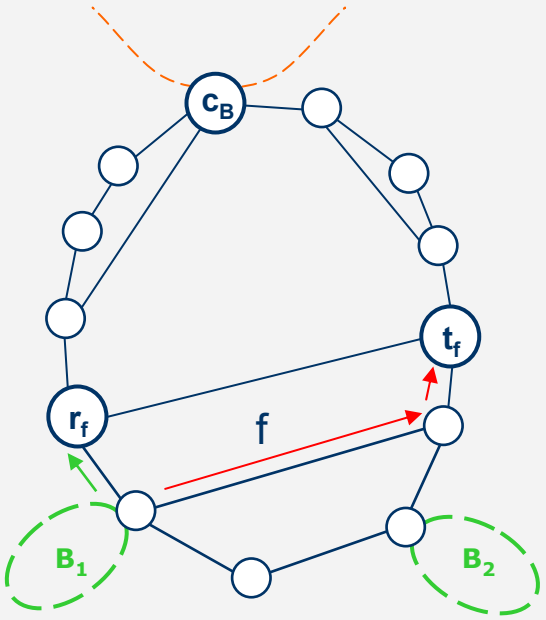
Preprocessing

For a given representation tree T_H rooted at block H

In addition, we compute quantities associated to the edges of G:

$S_{r_f t_f}$ the sum of the weighted distances to r_f (or to t_f) from all the vertices of G belonging to $V(T_B)\setminus[V(B)\setminus V(r_f,t_f)]$ that cannot be definitively assigned to r_f or to t_f (vertices unassigned in f)

$W_{r_f t_f}$ the sum of the weights of the vertices of G belonging to $V(T_B)\setminus[V(B)\setminus V(r_f,t_f)]$ that are unassigned in f



Actually, we compute the quantities W_{uv} and S_{uv} *for all* the edges $(u,v) \in E$

All the vertices of block B_1 are assigned to r_f in f

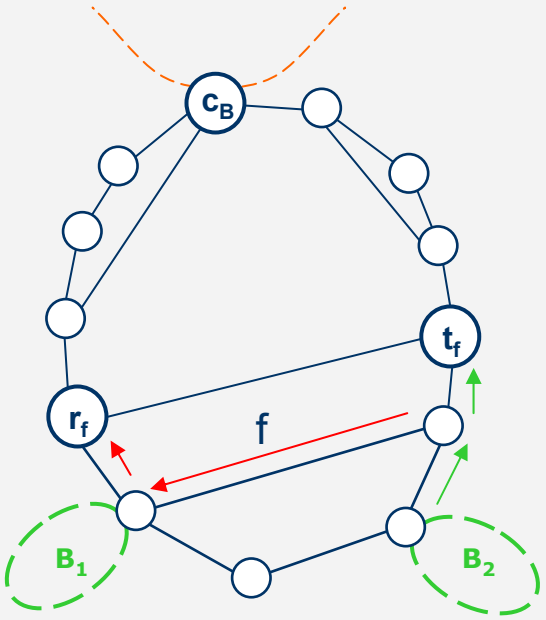
Preprocessing

For a given representation tree T_H rooted at block H

In addition, we compute quantities associated to the edges of G:

$S_{r_f t_f}$ the sum of the weighted distances to r_f (or to t_f) from all the vertices of G belonging to $V(T_B)\setminus[V(B)\setminus V(r_f,t_f)]$ that cannot be definitively assigned to r_f or to t_f (vertices unassigned in f)

$W_{r_f t_f}$ the sum of the weights of the vertices of G belonging to $V(T_B)\setminus[V(B)\setminus V(r_f,t_f)]$ that are unassigned in f



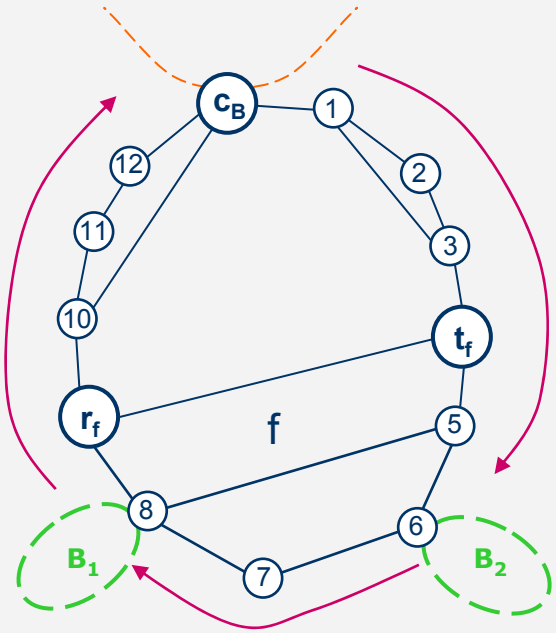
Actually, we compute the quantities W_{uv} and S_{uv} *for all* the edges $(u,v) \in E$

All the vertices of block B_2 are assigned to t_f in f

Preprocessing

For a given representation tree T_H rooted at block H

The preprocessing phase starts at the leaves of T_H and proceeds bottom up block by block.

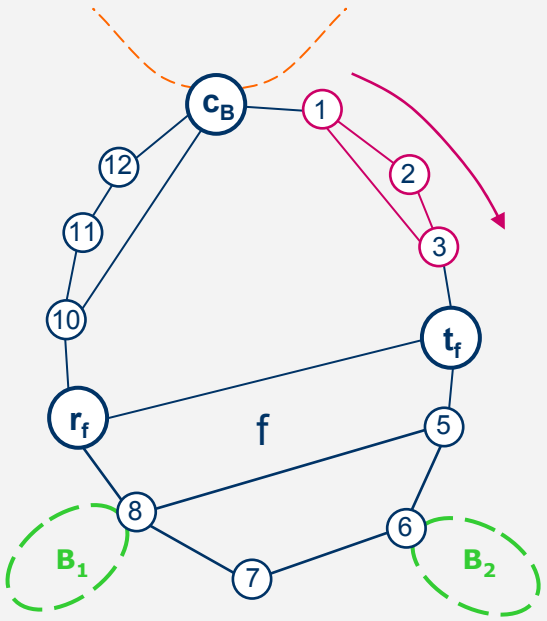


At each block B it computes the above quantities by visiting the vertices of B following the numbers from 1 to |V(B)|.

Preprocessing

For a given representation tree T_H rooted at block H

The preprocessing phase starts at the leaves of T_H and proceeds bottom up block by block.



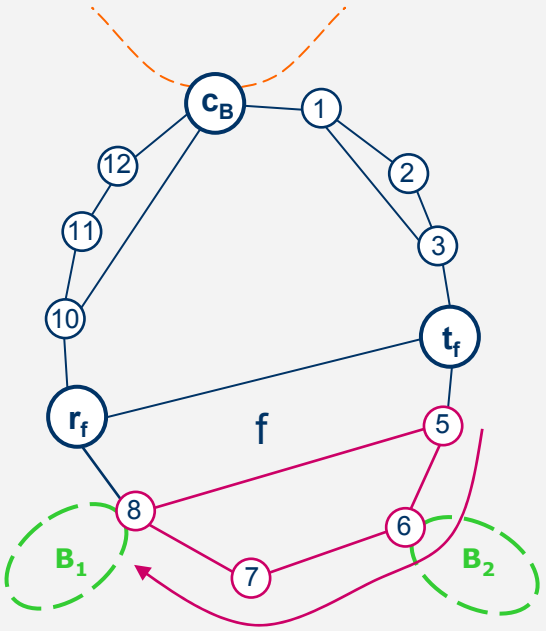
At each block B it computes the above quantities by visiting the vertices of B following the numbers from 1 to |V(B)|.

Actually, this corresponds to the visit of the faces of B from 1 to F_B.

Preprocessing

For a given representation tree T_H rooted at block H

The preprocessing phase starts at the leaves of T_H and proceeds bottom up block by block.



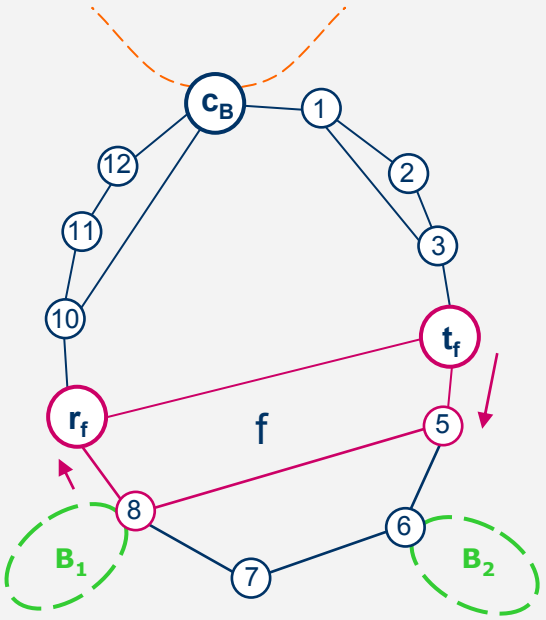
At each block B it computes the above quantities by visiting the vertices of B following the numbers from 1 to |V(B)|.

Actually, this corresponds to the visit of the faces of B from 1 to F_B.

Preprocessing

For a given representation tree T_H rooted at block H

The preprocessing phase starts at the leaves of T_H and proceeds bottom up block by block.



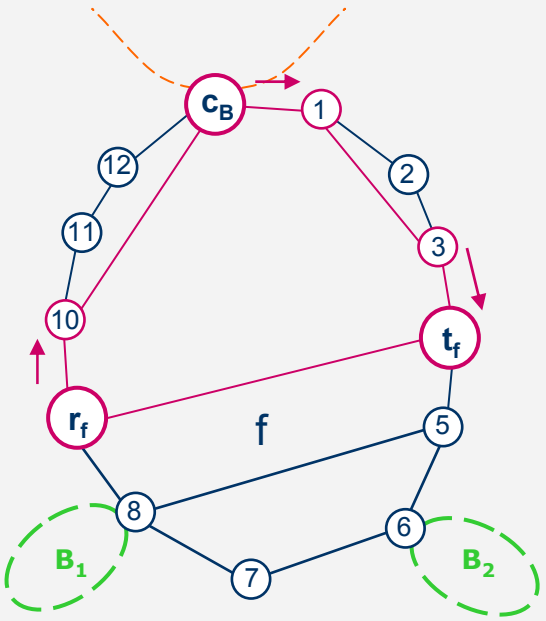
At each block B it computes the above quantities by visiting the vertices of B following the numbers from 1 to |V(B)|.

Actually, this corresponds to the visit of the faces of B from 1 to F_B.

Preprocessing

For a given representation tree T_H rooted at block H

The preprocessing phase starts at the leaves of T_H and proceeds bottom up block by block.



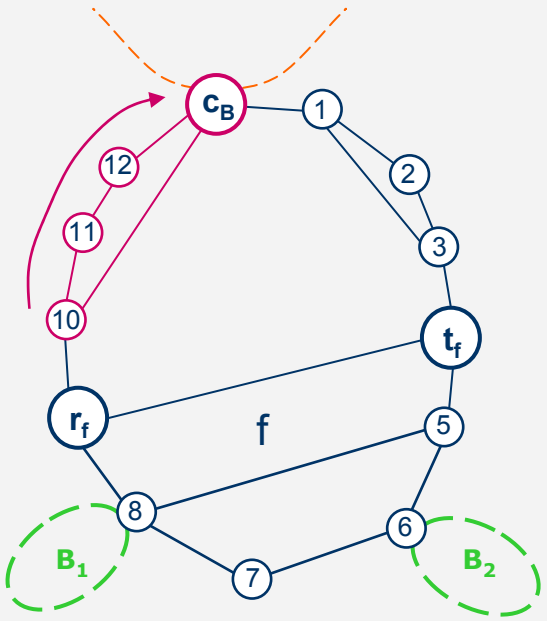
At each block B it computes the above quantities by visiting the vertices of B following the numbers from 1 to |V(B)|.

Actually, this corresponds to the visit of the faces of B from 1 to F_B.

Preprocessing

For a given representation tree T_H rooted at block H

The preprocessing phase starts at the leaves of T_H and proceeds bottom up block by block.



At each block B it computes the above quantities by visiting the vertices of B following the numbers from 1 to |V(B)|.

Actually, this corresponds to the visit of the faces of B from 1 to F_B.

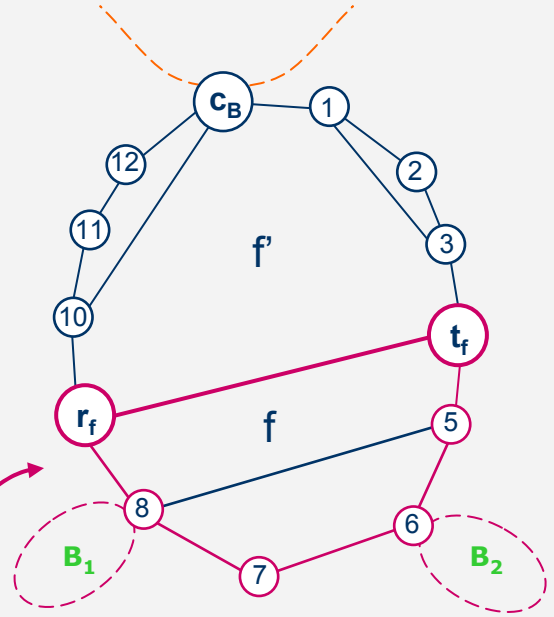
Preprocessing

For a given representation tree T_H rooted at block H

Best path from H to v in f
any path that, among all the paths starting at some vertex in H and ending in a vertex v belonging to face f, has minimum *distsum*.
We denote distsum of a best path from H to v in f by $D_H(v,f)$.

$V(r_f,t_f)$ is the set of vertices v in B such that $t_f \leq v \leq r_f$

Saving
During the preprocessing, in face f we compute another quantity that we denote by $Sav(r_f,t_f)$.
It corresponds to the sum of the weighted distances of all the vertices in $V(TB)\setminus[V(B)\setminus V(r_f,t_f)]$ to (r_f,t_f).

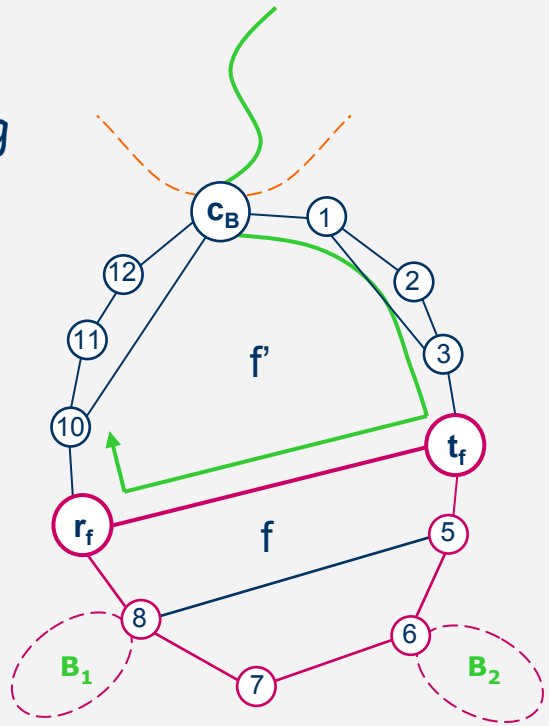


Preprocessing

For a given representation tree T_H rooted at block H

$Sav(r_f,t_f)$ will be used by the algorithm when in f' > f it will evaluate the *distsum* of a best path from H to u in f', with $u \neq r_f, t_f$, passing through vertices r_f and t_f.

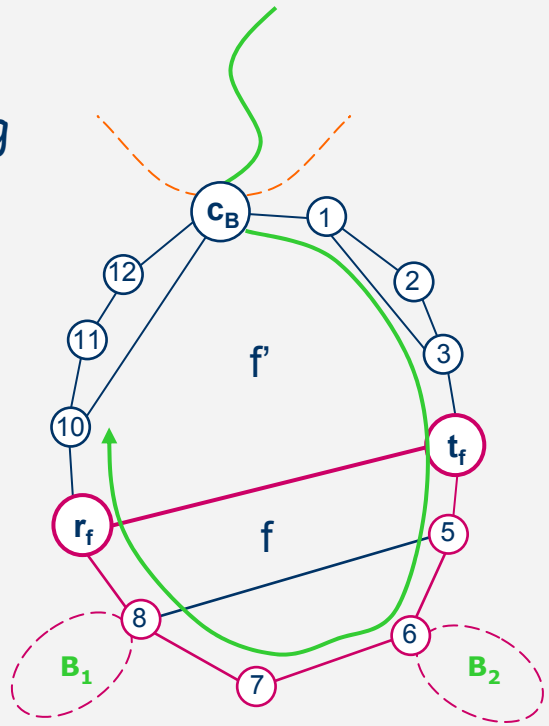
Consider for example vertex 10 in face f'. When in face f' the algorithm evaluates the path from H to 10 passing through r_f and t_f, it must be aware of the possibility of a path that passes through all the vertices in $V(TB)\setminus[V(B)\setminus V(r_f,t_f)]$ instead of the one that passes through the chord (r_f,t_f). The *distsum* of the former path is just equal to the *distsum* of the latter minus $Sav(r_f,t_f)$.



Preprocessing

For a given representation tree T_H rooted at block H

$Sav(r_f,t_f)$ will be used by the algorithm when in $f' > f$ it will evaluate the *distsum* of a best path from H to u in f', with $u \neq r_f, t_f$, passing through vertices r_f and t_f.



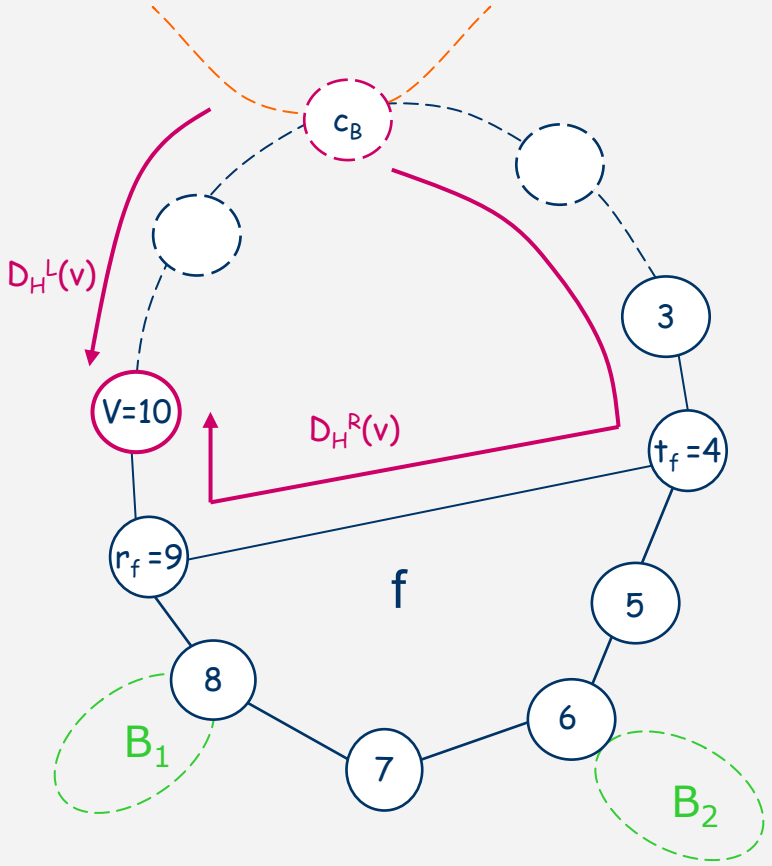
Consider for example vertex 10 in face f'. When in face f' the algorithm evaluates the path from H to 10 passing through r_f and t_f, it must be aware of the possibility of a path that passes through all the vertices in $V(TB)\setminus[V(B)\setminus V(r_f,t_f)]$ instead of the one that passes through the chord (r_f,t_f). The *distsum* of the former path is just equal to the *distsum* of the latter minus $Sav(r_f,t_f)$.

The preprocessing phase applied to T_H runs in O(n) time.

Formulas

For a given representation tree T_H rooted at block H

The complexity result is due to the fact that we exploit the structure of our outerplanar graph G in order to arrange an efficient computation of the *distsum* $D_H(u)$, for all u in each face f.



Actually, since every vertex v lies in a face of G, when the algorithm visits face f, for each v in f, it computes the following two quantities:

$D_H^L(v)$ the minimum *distsum* of a path from H to u that visits f in a counterclockwise order (counterclockwise path);

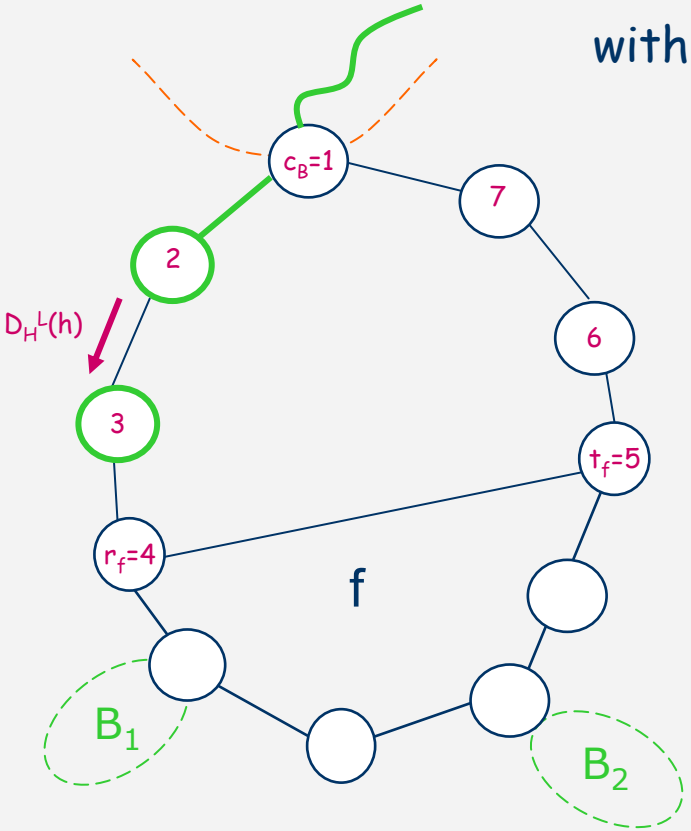
$D_H^R(v)$ the minimum *distsum* of a path from H to u that visits f in a clockwise order (clockwise path).

Formulas

For a given representation tree T_H rooted at block H

$$D_H^L(h) = \begin{cases} D^L(h-1) - [W^R(h) - W^R(m/2, (m/2)+1)] - Sav(h-1,h), & \text{if m is even} \\ D^L(h-1) - [W^R(h) - W^R(\lceil m/2 \rceil)] - Sav(h-1,h), & \text{if m is odd} \end{cases}$$

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



where:

n_f is the number of vertices in f

$W^R(h)$ is a cumulative weight computed clockwise starting from vertex 6 by summing the the weights of visited vertices

$m = (n_f+h)$

Notice that during the algorithm the vertices are locally re-labeled face by face.