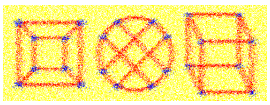

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 26, no. 4, pp. 589–606 (2022)
DOI: 10.7155/jgaa.00610

Non-Crossing Shortest Paths in Undirected Unweighted Planar Graphs in Linear Time

Lorenzo Balzotti[1] *Paolo G. Franciosa*[1]

[1]Dipartimento di Scienze Statistiche, Sapienza Università di Roma, p.le Aldo Moro 5, 00185 Roma, Italy

Submitted:	July 2022	Reviewed:	October 2022
Revised:	October 2022	Accepted:	December 2022
Final:	December 2022	Published:	December 2022
Article type:	Regular paper	Communicated by:	G. Liotta

Abstract. Given a set of terminal pairs on the external face of an undirected unweighted planar graph, we give a linear-time algorithm for computing the union of non-crossing shortest paths joining each terminal pair, if such paths exist. This allows us to compute distances between each terminal pair, within the same time bound. We also give a novel concept of *incremental shortest path* subgraph of a planar graph, i.e., a partition of the planar embedding in subregions that preserve distances, that can be of interest itself.

1 Introduction

The problem of computing shortest paths in planar graphs arises in application fields such as intelligent transportation system (ITS) and geographic information system (GIS) [27, 47], route planning [6, 21, 40], logistic [36], traffic simulations [2] and robotics [28].

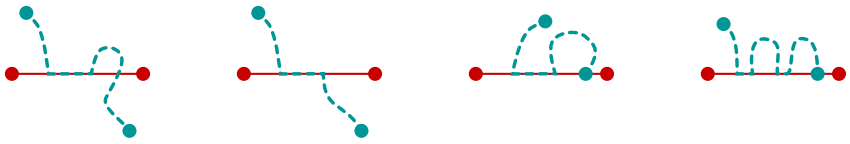
We are given a plane graph $G = (V, E)$, i.e., a planar graph with a fixed planar embedding, where V is a set of n vertices and E is a set of edges, with $|E| = O(n)$. We are also given a set of k terminal pairs $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$ lying on the external face of G. The non-crossing shortest paths problem (NCSP problem) consists in computing the union of k non-crossing shortest paths in G, each joining a terminal pair (s_i, t_i), provided that such non-crossing paths exist (they exist if and only if the terminal pairs are *well-formed*, see Subsection 2.2).

Two paths in a plane graph are *non-crossing* if the (undirected) curves they describe in the graph embedding do not cross each other, non-crossing paths may share vertices and/or edges

A preliminary version of this paper appeared in [5].

E-mail addresses: lorenzo.balzotti@uniroma1.it (Lorenzo Balzotti) paolo.franciosa@uniroma1.it (Paolo G. Franciosa)

or darts, see Figure 1. This property obviously depends on the embedding of the graph; a combinatorial definition of non-crossing paths can be based on the *Heffter-Edmonds-Ringel rotation principle* [22]. Non-crossing shortest paths in plane graphs are studied to optimize VLSI layout [9, 33, 34].



(a) (b) (c) (d)

Figure 1: paths in (a) and (b) are crossing, while paths in (c) and (d) are non-crossing.

State of the art and related work Takahashi *et al.* [44] solved the NCSP problem in a non-negative edge-weighted plane graph in $O(n \log k)$ time (actually, in their paper the time complexity is $O(n \log n)$, that can easily reduced to $O(n \log k)$ by applying the planar single source shortest path algorithm by Henzinger *et al.* [25]). Their result is improved by Steiger in $O(n \log \log k)$ time [43], exploiting the algorithm by Italiano *et al.* [26]. These two algorithms maintain the same time complexity also in the unweighted case.

Our problem fits into a wider context of computing many distances in planar graphs. In the positive weighted case, the all pairs shortest paths (APSP) problem is solved by Frederickson in $O(n^2)$ time [19], while the single source shortest paths (SSSP) problem is solved in linear time by Henzinger *et al.* [25]. The best known algorithm for computing many distances in planar graphs is due to Gawrychowski *et al.* [20] and it allows us to compute the distance between any two vertices in $O(\log n)$ time after a preprocessing requiring $O(n^{3/2})$ time. In the plane unweighted case, SSSP trees rooted at vertices in the external face can be computed in linear time as in [15], as already mentioned. More results on many distances problem can be found in [10, 11, 13, 18, 37, 38].

If we are interested in distances from any vertex in the external face to any other vertex, then we can use Klein's algorithm [29] that, with a preprocessing of $O(n \log n)$ time, answers to each distance query in $O(\log n)$ time.

Kowalik and Kurowski [31] deal with the problem of deciding whether any two query vertices of an unweighted planar graph are closer than a fixed constant k. After a preprocessing of $O(n)$ time, their algorithm answers in $O(1)$ time, and, if so, a shortest path between them is returned.

Non-crossing shortest paths are also used to compute max-flow in undirected planar graphs [23, 24, 41]. In particular, they are used to compute the vitality of edges and vertices with respect to the max-flow [1, 4].

Balzotti and Franciosa [3] show that, given the union of a set of non-crossing shortest paths in a plane graph, the lengths of each shortest path can be computed in linear time. This improves the result of [44], that can only be applied when the union of the shortest paths is a forest.

Wagner and Weihe [46] present an $O(n)$ time algorithm for finding edge-disjoint (not necessarily shortest) paths in an undirected plane graph such that each path connects two specified vertices on the external face of the graph.

In a geometrical setting Papadopoulou [39] finds the set of k non-crossing shortest paths between k terminal pairs of points on the boundary of a simple polygon with n vertices in $O(n + k)$ time.

Eriksson-Bique *et al.* [17] study the problem of computing shortest paths in a two-dimensional environment with polygonal obstacles.

More results on disjoint shortest paths for general graphs can be found in [7, 8, 14, 35] and in [12, 30, 42] for the planar case.

Our results In this paper, we solve the NCSP problem on undirected unweighted plane graphs in $O(n)$ time. We improve, in the unweighted case, the results in [43, 44]. Given the union of non-crossing shortest paths we can also compute distances between terminal pairs in total linear time by algorithm in [3].

Our algorithm relies on two main results:

- an algorithm due to Eisenstat and Klein [15], that gives in $O(n)$ time an implicit representation of a sequence of shortest path (SSSP) trees in an undirected unweighted plane graph G, where each tree is rooted at a vertex of the external face of G. Note that, if we want to compute shortest paths from the implicit representation of shortest path trees given in [15], then we spend $\Theta(kn)$ time; this happens when all k shortest paths share a subpath of $\Theta(n)$ edges.

- the novel concept of *incremental shortest paths (ISP) subgraph* of a plane graph G. We show that an ISP subgraph of G partitions the embedding of G into *distance preserving* regions, i.e., for any two vertices a, b in G lying in the same region R it is always possible to find a shortest path in G joining a and b that is contained in R.

Improved results We specialize the problem of finding k non-crossing shortest paths in [44] to the unweighted case, decreasing the time complexity from $O(n \log k)$ to $O(n)$ (for every k). Therefore, in the case of unweighted graphs we improve the results in [16, 32, 45].

Erickson and Nayyeri [16] generalized the work in [44] to the case in which the k terminal pairs lie on h face boundaries. They prove that k non-crossing paths, if they exists, can be found in $2^{O(h^2)} n \log k$ time. Applying our results, if the graph is unweighted, then the time complexity decreases to $2^{O(h^2)} n$.

The same authors of [44] used their algorithm to compute k non-crossing rectilinear paths with minimum total length in a plane graph with r obstacles [45]. They found such paths in $O(n \log n)$ time, where $n = r + k$, which reduces to $O(n)$ time if the graph is unweighted by using our results.

Kusakari *et al.* [32] showed that a set of non-crossing forests in a plane graph can be found in $O(n \log n)$ time, where two forests F_1 and F_2 are *non-crossing* if for any pair of paths $p_1 \subseteq F_1$ and $p_2 \subseteq F_2$, p_1 and p_2 are non-crossing. With our results, if the graph is unweighted, then the time complexity becomes linear.

Our approach We represent the structure of terminal pairs by a partial order called *genealogy tree* as in [44]. We introduce a new class of graphs, ISP subgraphs, that partition a plane graph into regions that preserve distances. Our algorithm is split in two parts.

In the first part we use Eisenstat and Klein's algorithm [15] that gives a sequence of shortest path trees rooted at the vertices of the external face. We choose some specific shortest paths from each tree to obtain a sequence of ISP subgraphs $X_1, \ldots X_k$. By using the distance preserving property of regions generated by ISP subgraphs, we prove that X_i contains a shortest s_i-t_i path, for all $i \in \{1, \ldots, k\}$.

In the second part of our algorithm, we extract from each X_i a shortest s_i-t_i path and we obtain a set of non-crossing shortest paths that is our goal. In this part we strongly use the partial order given by the genealogy tree.

Structure of the paper After giving some definitions in Section 2, in Section 3 we explain the main theoretical novelty. In Section 4 first we resume Eisenstat and Klein's algorithm in Subsection 4.1, then in Subsections 4.2 and 4.3 we show the two parts of our algorithm, and we prove the whole computational complexity. Conclusions are given in Section 5.

2 Definitions

Let G be a plane graph, we denote by f_G^∞ (or simply f^∞) its unique external face, it will be also referred to as the *external* face of G. Given a face f of G we denote by ∂f its boundary cycle. Topological and combinatorial definitions of planar graph, embedding and face can be found in [22].

We recall standard union and intersection operators on graphs.

Definition 1 *Given two undirected (or directed) graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$, we define the following operations and relations:*

- $G \cup H = (V(G) \cup V(H), E(G) \cup E(H))$,
- $G \cap H = (V(G) \cap V(H), E(G) \cap E(H))$,
- $G \subseteq H \iff V(G) \subseteq V(H)$ *and* $E(G) \subseteq E(H)$,
- $G \setminus H = (V(G), E(G) \setminus E(H))$.

Given an undirected (resp., directed) graph $G = (V(G), E(G))$, given an edge (resp., dart) e and a vertex v we write, for short, $e \in G$ in place of $e \in E(G)$ and $v \in G$ in place of $v \in V(G)$.

We denote by uv the edge whose endpoints are u and v and we denote by $\overrightarrow{uv}$ the dart from u to v. For each dart $\overrightarrow{uv}$ we define $\text{rev}[\overrightarrow{uv}] = \overrightarrow{vu}$, $\text{head}[\overrightarrow{uv}] = v$ and $\text{tail}[\overrightarrow{uv}] = u$. For each vertex $v \in V(G)$ we define the *degree of* v as $deg(v) = |\{e \in E(G) \mid v \text{ is an endpoint of } e\}|$.

For each $\ell \in \mathbb{N}$ we denote by $[\ell]$ the set $\{1, \ldots, \ell\}$.

Given a (possibly not simple) cycle C, we define the *region bounded by* C, denoted by R_C, as the maximal subgraph of G whose external face has C as boundary.

2.1 Paths and non-crossing paths

Given a directed path p we denote by $\overline{p}$ its undirected version, in which each dart $\overrightarrow{ab}$ is replaced by edge ab; moreover, we denote by $\text{rev}[p]$ its reverse version, in which each dart $\overrightarrow{ab}$ is replaced by dart $\overrightarrow{ba}$.

We say that a path p is an *a-b path* if its extremal vertices are a and b; clearly, if p is a directed path, then p starts in a and it ends in b. Moreover, given $i \in [k]$, we denote by *i-path* an s_i-t_i path, where (s_i, t_i) is one of the terminal pairs on the external face.

Given an a-b path p and a b-c path q, we define $p \circ q$ as the (possibly not simple) a-c path obtained by the union of p and q.

Let p be a simple path and let $a, b \in V(p)$. We denote by $p[a, b]$ the subpath of p with extremal vertices a and b.

We denote by $\omega(p)$ the length of a path p of a general positive weighted graph G. If G is unweighted, then we denote the length of p as $|p|$, that is the number of edges.

2.2 Genealogy tree

W.l.o.g., we assume that terminal pairs are distinct, i.e., there is no pair $i, j \in [k]$ such that $\{s_i, t_i\} = \{s_j, t_j\}$. Let γ_i be the path in f^∞ that goes clockwise from s_i to t_i, for $i \in [k]$. We also assume that pairs $\{(s_i, t_i)\}_{i\in[k]}$ are *well-formed*, i.e., for all $j, \ell \in [k]$ either $\gamma_j \subseteq \gamma_\ell$ or $\gamma_j \supseteq \gamma_\ell$ or γ_j and γ_ℓ have no common edges; otherwise it can be easily seen that it is not possible to find a set of k non-crossing paths joining terminal pairs. This property can be easily verified in linear time, since it corresponds to checking that a string of parentheses is balanced, and it can be done by a sequential scan of the string.

We define here a partial ordering as in [3, 44] that represents the inclusion relation between γ_i's. This relation intuitively corresponds to an *adjacency* relation between non-crossing shortest paths joining each pair. Choose an arbitrary i^* such that there are neither s_j nor t_j, with $j \neq i^*$, walking on f^∞ from s_{i^*} to t_{i^*} (either clockwise or counterclockwise), and let e^* be an arbitrary edge on that walk. For each $j \in [k]$, we can assume that $e^* \notin \gamma_j$, indeed if it is not true, then it suffices to switch s_j with t_j; in this way $\gamma_j \subseteq \gamma_{i^*}$ for every $j \in [k]$.

We say that $i \preceq j$ if $\gamma_i \subseteq \gamma_j$. We define the *genealogy tree* T_G of a set of well-formed terminal pairs as the transitive reduction of poset $([k], \preceq)$. By the above choice, i^* is the root of the genealogy tree, and there are as many possible genealogy trees as many leaves of each genealogy tree. W.l.o.g., we assume that $i^* = 1$.

If $i \preceq j$, then we say that i is a *descendant* of j and j is an *ancestor* of i. Moreover, we say that j is the *parent of* i, and we write $p(i) = j$, if $i \preceq j$ and there is no r such that $i \preceq r$ and $r \preceq j$. Figure 2 shows a set of well-formed terminal pairs, and the corresponding genealogy tree for $i^* = 1$.

From now on, in all figures we draw f^∞ by a solid light grey line. W.l.o.g., we assume that the external face is a simple cycle and that G is a biconnected plane graph. Indeed, if there is an articulation point z in G, then it suffices to solve an instance for each component. The solution to the original problem is the union of the solutions on the two components by observing that if s_i and t_i are in two distinct components, then every i-path passes through z.

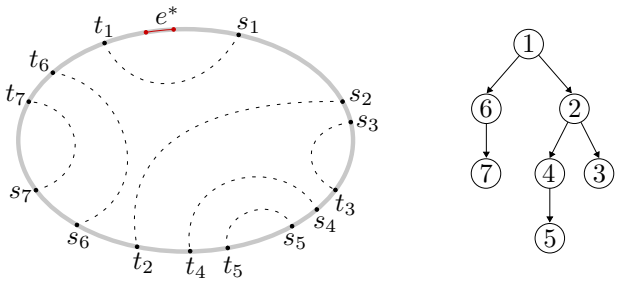


Figure 2: on the left a set of well-formed terminal pairs. Any value in $\{1, 3, 5, 7\}$ can be chosen as i^*. If we choose $i^* = 1$, then we obtain the genealogy tree on the right.

3 ISP subgraphs

In this section we introduce the concept of *incremental shortest paths (ISP) subgraph* of a plane graph G, that is a subgraph incrementally built by adding a sequence of shortest paths in G starting from f^∞ (see Definition 2). The interest towards ISP subgraphs is due to the fact that for any two vertices a, b in G lying in a same face f of the ISP subgraph there is always a shortest path in G

joining a and b contained in f (boundary included). All the results of this section hold for positive weighted graphs, where the length of a path is the sum of edge weights instead of the number of edges.

This is the main novel result of this paper, that allows us to prove that, in order to build the union of shortest paths joining terminal pairs, we can start from the union of some of the shortest paths computed by the algorithm in [15].

Definition 2 *A graph X is an* incremental shortest paths (ISP) subgraph *of a positive weighted undirected plane graph G if $X = X_r$, where X_1, X_2, ..., X_r is a sequence of subgraphs of G built in the following way: $X_1 = f^\infty$ and $X_i = X_{i-1} \cup p_i$, where p_i is a shortest x_i-y_i path in G with $x_i, y_i \in X_{i-1}$.*

Remark 1 *All degree one vertices of an ISP subgraph of G are in f^∞.*

We define now operator $\downarrow$, that given a path π and a cycle C, in case π crosses C, replaces some subpaths of π by some portions of C, as depicted in Figure 3(b). We observe that $\pi \downarrow \partial f$ could be not a simple path even if π is.

Definition 3 *Let C be a cycle in a positive weighted undirected plane graph G. Let a, b be two vertices in R_C and let π be a simple a-b path. In case $\pi \subseteq R_C$ we define $\pi \downarrow C = \pi$. Otherwise, let $(v_1, v_2, \ldots, v_{2r})$ be the ordered subset of vertices of π that satisfies the following: $\pi[a, v_1] \subseteq R_C$, $\pi[v_{2r}, b] \subseteq R_C$, $\pi[v_{2i-1}, v_{2i}]$ and R_C have no common edges and $\pi[v_{2i}, v_{2i-1}] \subseteq R_C$, for all $i \in [r]$. For each $i \in [r]$, let μ_i be the v_{2i-1}-v_{2i} path on C such that the region bounded by $\mu_i \circ \pi[v_{2i-1}, v_{2i}]$ does not contain R_C. We define $\pi \downarrow C = \pi[a, v_1] \circ \mu_1 \circ \pi[v_2, v_3] \circ \mu_2 \ldots \circ \pi[v_{2r-2}, v_{2r-1}] \circ \mu_r \circ \pi[v_{2r}, b]$.*

Definition 2 and Definition 3 are depicted in Figure 3.

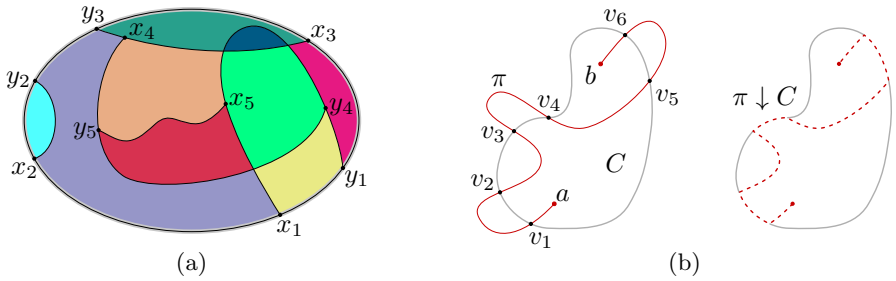


Figure 3: (a) an ISP subgraph X of G; extremal vertices x_i, y_i of p_i are drawn, for $i \in [5]$. Different faces of X have different colors. An example of Definition 3 is given in (b).

Note that every face f of an ISP subgraph X of G induces the region R_{∂_f} in G, and this region may contain vertices in G that are not in X. In the following theorem we show that, given any face f of an ISP subgraph X of G, every path π in G whose extremal vertices are in $R_{\partial f}$ is not shorter than $\pi \downarrow \partial f$.

Theorem 1 *Let X be an ISP subgraph of a positive weighted undirected plane graph G. Let f be any face of X, and let a, b be two distinct vertices in $R_{\partial f}$. For any a-b path π we have $\omega(\pi \downarrow \partial f) \leq \omega(\pi)$.*

Proof: Let $\{X_i\}_{i\in[r]}$ be the sequence of ISP subgraphs such that $X = X_r$, and let p_i be the path that builds X_i from X_{i-1}. We assume that p_i has no vertices in X_{i-1} other than its endpoints x_i and y_i, otherwise we can split p_i on intersections with X_{i-1} and repeatedly apply the same proof to each portion of p_i. We prove the thesis by induction on r for every choice of a face f of X_r, $a, b \in R_{\partial f}$ and a-b path π.

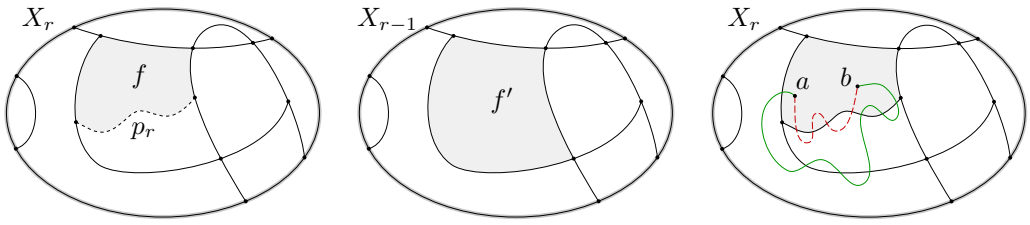
In the base case, where $r = 1$, X_r is equal to f^∞ by Definition 2, thus for every path π we trivially have that $\pi \downarrow \partial f^\infty = \pi$. Hence, $\omega(\pi \downarrow \partial f) = \omega(\pi)$ and the thesis holds. Let us assume that the thesis is true for $r-1$ and let us prove it for r.

Let f be a face of X_r and let f' be the unique face of X_{r-1} such that $f \subset f'$ (Figure 4(a) and Figure 4(b) show faces f and f', respectively). Let $a, b \in V(R_{\partial f})$ and let π be an a-b path. Three cases may occur:

case $\pi \subseteq R_{\partial f}$**:** the thesis trivial holds, since $\pi \downarrow \partial f = \pi$;

case $\pi \subseteq R_{\partial f'}$ **and** $\pi \not\subseteq R_{\partial f}$**:** since $\pi \subseteq R_{\partial f'}$ and $\pi \not\subseteq R_{\partial f}$, then π crosses p_r an even number of times, thus $\pi \downarrow \partial f$ is not longer than π, since some subpaths of π have been replaced by subpaths of p_r with the same extremal vertices and p_r is a shortest path (see Figure 4(c) where π is the red dashed path);

case $\pi \not\subseteq R_{\partial f'}$**:** since $f \subseteq f'$, it is easy to see that $\pi \downarrow \partial f = (\pi \downarrow \partial f') \downarrow \partial f$. Let us consider $\pi' = \pi \downarrow \partial f'$. By induction, it holds that $\omega(\pi') \leq \omega(\pi)$. We observe now that $\pi' \subseteq R_{\partial f'}$ and $\pi' \not\subseteq R_{\partial f}$, hence the previous case applies, showing that $\omega(\pi' \downarrow \partial f) \leq \omega(\pi')$. Finally, the two previous inequalities imply $\omega(\pi \downarrow \partial f) = \omega((\pi \downarrow \partial f') \downarrow \partial f) = \omega(\pi' \downarrow \partial f) \leq \omega(\pi') \leq \omega(\pi)$ (see Figure 4(c) where π is the green continue path). □



(a) a face f in X_r

(b) a face f' in X_{r-1}

(c) two examples of π in the second case (dashed red) and third case (continuous green)

Figure 4: in (a) and (b) faces f and f' build on the ISP subgraph in Figure 3(a). In (c) we depict the second and third case of the proof of Theorem 1.

We can state now the main property of ISP subgraphs.

Corollary 2 *Let X be an ISP subgraph of a positive weighted undirected plane graph G and let f be a face of X. For every $a, b \in R_{\partial f}$ there exists a shortest a-b path of G contained in $R_{\partial f}$.*

4 Our algorithm

We summarize in Subsection 4.1 the result of Eisenstat and Klein's paper [15], that deals with the multiple-source shortest paths problem. For the sake of clarity, we split our algorithm in two parts:

- in Subsection 4.2 we introduce algorithm `NCSPsupergraph`, that builds a sequence $\{X_i\}_{i\in[k]}$ of subgraphs of G such that X_k contains a shortest path for each terminal pair, and it possibly contains some extra edges. We anticipate that $X_i \cup f^\infty$ is an ISP subgraph of G, for all $i \in [k]$.
- in Subsection 4.3 we present algorithm `NCSPunion` that, by using the sequence of graphs $\{X_i\}_{i\in[k]}$ found by algorithm `NCSPsupergraph`, builds a directed graph that is exactly the union of the shortest directed paths joining each terminal pair contained in the output of algorithm `NCSPsupergraph`.

4.1 Eisenstat and Klein's result

The algorithm in [15] takes as input an undirected unweighted plane graph G, where $v_1, v_2, \ldots, v_r$ is the sequence of vertices in the external face of G in clockwise order, and returns an implicit representation of a sequence of shortest path trees $\mathcal{T}_{v_i}$, for $i \in [r]$, where each $\mathcal{T}_{v_i}$ is rooted at v_i.

The sequence of trees $\mathcal{T}_{v_i}$, for $i \in [r]$, is represented by explicitly listing the darts in $\mathcal{T}_{v_1}$, and listing the darts that are added to transform $\mathcal{T}_{v_i}$ into $\mathcal{T}_{v_{i+1}}$, for $1 < i \leq r$ (for each added dart from x to y, the unique dart that goes to y in $\mathcal{T}_{v_i}$ is deleted; with the only two exceptions of the added dart leading to v_i, and the deleted dart leading to v_{i+1}). Hence, the output of their algorithm is $\mathcal{T}_{v_1}$ and a sequence of sets of darts. A key result in [15] shows that if a dart d appears in $\mathcal{T}_{v_{i+1}} \setminus \mathcal{T}_{v_i}$, then d cannot appear in any $\mathcal{T}_{v_{j+1}} \setminus \mathcal{T}_{v_j}$, for $j > i$. Thus the implicit representation of the sequence of shortest path trees has size $O(n)$. This representation can be computed in $O(n)$ time.

4.2 Algorithm `NCSPsupergraph`

Algorithm `NCSPsupergraph` builds a sequence $\{X_i\}_{i\in[k]}$ of subgraphs of G by using the sequence of shortest path trees given by Eisenstat and Klein's algorithm. We point out that we are not interested in the shortest path trees rooted at every vertex of f^∞, but we only need the shortest path trees rooted at s_i's. So, we define T_i as the shortest path tree rooted at s_i, for $i \in [k]$, i.e., $T_i = \mathcal{T}_{s_i}$. We denote by $T_i[v]$ the path in T_i from s_i to v.

The algorithm starts by computing the first subgraph X_1, that is just the undirected 1-path in T_1, i.e., $\overline{T_1[t_1]}$ (we recall that all T_i's trees given by algorithm in [15] are rooted directed trees, thus $\overline{T_1}$ is the undirected version of T_1). Then the sequence of subgraphs X_i, for $i = 2, \ldots, k$ is computed by adding some undirected paths extracted from the shortest path trees T_i's defined by Eisenstat and Klein's algorithm.

We define the set $H_i \subseteq X_i$ of vertices h such that at least one dart d is added while transforming T_{i-1} into T_i such that $\text{head}[d] = h$. Hence, H_i is the set of vertices of X_i whose parent in T_i differs from the parent in T_{i-1}. At iteration i, we add path $\overline{T_i[h]}$ to X_i, for each h in H_i.

Lemma 1 *Algorithm `NCSPsupergraph` has $O(n)$ time complexity.*

Proof: Eisenstat and Klein's algorithm requires $O(n)$ time, implying that the H_i's and the T_i's can be found in $O(n)$ time. Algorithm `NCSPsupergraph` visits each edge of G at most $O(1)$ times because it builds X_i without visiting edges in X_{i-1} (in Line 7, $\overline{T_i[h]}$ can be found by starting in h and by walking backwards on T_i until a vertex of X_i is found). The thesis follows. □

Figure 5 shows how algorithm `NCSPsupergraph` builds X_4 starting from X_3. Starting from X_3 in Figure 5(a), Figure 5(b) shows the darts whose head is in H_4. Consider the unique dart d whose head is the vertex x: we observe that $\overline{d}$ is already in X_3, this happens because $\text{rev}[d] \in T_3[t_3]$. Indeed, it is possible that at iteration i some portions of some undirected paths that we add in

Algorithm `NCSPsupergraph`**:**

Input: an undirected unweighted plane graph G and k well-formed terminal pairs $\{(s_i, t_i)\}_{i\in[k]}$ on the external face of G

Output: an undirected graph X_k that contains a set of non-crossing paths $P = \{\pi_1, \ldots, \pi_k\}$, where π_i is a shortest s_i-t_i path, for $i \in [k]$

1 Compute a shortest path tree T_1 rooted at s_1;
2 $X_1 = \overline{T_1[t_1]}$;
3 **for** $i = 2, \ldots, k$ **do**
4 $\quad X_i = X_{i-1}$;
5 $\quad$ Compute T_i from T_{i-1} by the algorithm by Eisenstat and Klein [15];
6 $\quad$ Compute the set H_i of vertices of X_i whose parent in T_i differs from the parent in T_{i-1};
7 $\quad$ For all $h \in H_i$, $X_i = X_i \cup \overline{T_i[h]}$;
8 $\quad$ Let η_i be the undirected path on T_i that starts in t_i and walks backwards until a vertex in X_i is reached;
9 $\quad X_i = X_i \cup \eta_i$;

Line 7 are already in X_{i-1}. Figure 5(c) highlights $\bigcup_{h\in H_4} T_4[h]$ and η_4, while in Figure 5(d) X_4 is drawn.

Subgraphs $\{X_i\}_{i\in[k]}$ built by algorithm `NCSPsupergraph`, together with f^∞, satisfy all the hypothesis of Theorem 1. Indeed, paths added in Line 7 and Line 9 are shortest paths in G joining vertices in X_{i-1}, thus fulfilling Definition 2. So, we exploit Theorem 1 to prove that X_i contains an i-path, for $i \in [k]$, and, in particular, X_k contains a set of non-crossing paths $P = \{\pi_1, \ldots, \pi_k\}$, where π_i is a shortest i-path, for $i \in [k]$. The main idea is to show that X_i contains an undirected path that has the same length as the shortest i-path found by the algorithm by Eisenstat and Klein. This is proved in Theorem 3.

Given a subgraph X of G, we say that an i-path p is the *leftmost i-path in X* if for every i-path $q \subseteq X$ it holds $R_{p\circ\gamma_i} \subseteq R_{q\circ\gamma_i}$.

We say that an undirected path p *always turns left* if p chooses the leftmost edge, w.r.t. the fixed embedding, in each vertex going from a to b, where a and b are the extremal vertices of p. Note that the leftmost a-b path is not necessarily the path that starts in a and always turns left until b is reached. To better understand the previous definition and the π_i's paths defined in the following theorem, we refer to Figure 6, showing paths $\pi_1, \pi_2, \pi_3, \pi_4$ built on graph X_4 in Figure 5(d).

Note that the following theorem describes a set of non-crossing shortest paths, but it does not solve our problem. Indeed if we extract π_i from X_i, for each $i \in [k]$, then we may spend $O(kn)$ time. We show how do it in linear time in the next section.

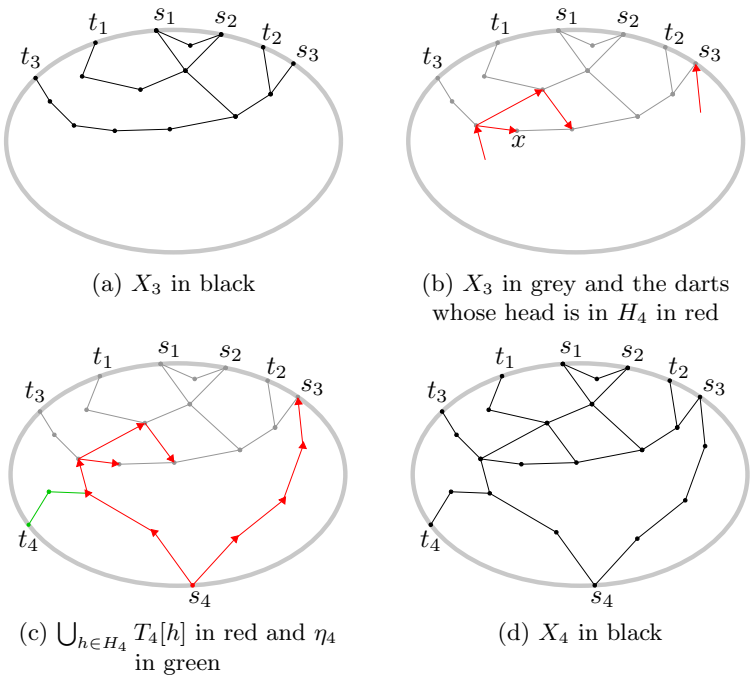
Theorem 3 *Let π_i be the undirected leftmost i-path in X_i, for $i \in [k]$. The following statements hold:*

3.(1) π_i is the s_i-t_i path in X_i that always turns left, for $i \in [k]$,

3.(2) π_i is a shortest i-path, for $i \in [k]$,

3.(3) for all $i, j \in [k]$, π_i and π_j are non-crossing.

Proof: We prove all the statements separately.



(a) X_3 in black

(b) X_3 in grey and the darts whose head is in H_4 in red

(c) $\bigcup_{h \in H_4} T_4[h]$ in red and η_4 in green

(d) X_4 in black

Figure 5: how algorithm `NCSPsupergraph` builds graph X_4 starting from X_3.

3.(1) For convenience, for each $i \in [k]$, let λ_i be the undirected path on X_i that starts in s_i and always turns left until it reaches either t_i or a vertex x of degree one in X_i; we observe that λ_i is well defined. We have to prove that $\lambda_i = \pi_i$.

Let $i \in [k]$. First, we observe that $s_i \in X_i$ because $s_{i-1} \in H_i$, thus, by Line 7, $\overline{T_i[s_{i-1}]} \subseteq X_i$. This implies $s_i \in X_i$ as we have claimed.

Let x be the extremal vertex of λ_i other than s_i; by Remark 1 and definition of λ_i, $x \in f^\infty$. Assume by contradiction that $x \neq t_i$. Two cases are possible: either $x \in V(f^\infty) \setminus V(\gamma_i)$ or $x \in V(\gamma_i) \setminus \{t_i\}$.

The first case cannot occur because Line 7 and Line 9 imply $\overline{T_i[t_i]} \subseteq X_i$, thus λ_i would cross η_i, absurdum. In the second case, let us assume by contradiction that $x \in V(\gamma_i) \setminus \{t_i\}$. Let $d \in \lambda_i$ be the dart such that $\mathrm{head}[d] = x$. By definition of λ_i, vertex x has degree one in X_i. By Line 2, Line 7 and Line 9, all vertices with degree one are equal to either s_ℓ or t_ℓ, for some $\ell \in [k]$, and this implies that there exists $j < i$ such that $x \in \{s_j, t_j\}$. This is absurdum because there is not s_j or t_j in $V(\gamma_i) \setminus \{s_i, t_i\}$ such that $j < i$. Hence λ_i is an i-path, and, by its definition, λ_i is the leftmost i-path in X_i. Therefore $\lambda_i = \pi_i$.

3.(2) We prove that π_i is a shortest i-path by using Theorem 1, indeed, $X_i \cup f^\infty$ is an ISP subgraph of G by construction. Let G' be the graph obtained from G by adding a dummy path q from s_i to t_i in f^∞ with high length (for example, $|q| = |E(G)|$). Let C be the cycle $\pi_i \circ q$. We observe that $\overline{T_i[t_i]} \downarrow C = \pi_i$ and C is the boundary of a face of G'. Thus, by Theorem 1, $|\pi_i| \leq |\overline{T_i[t_i]}|$. Since $\overline{T_i[t_i]}$ is a shortest path, then π_i is a shortest path in G', hence it also is a shortest path in G.

3.(3) Let us assume by contradiction that there exist $i, j \in [k]$ such that π_i and π_j are crossing, with $i < j$. Thus π_j has not turned always left in X_j, absurdum. □

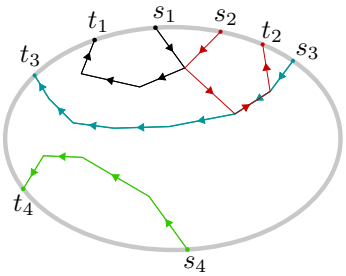


Figure 6: paths $\pi_1, \pi_2, \pi_3, \pi_4$ built on graph X_4 in Figure 5(d).

4.3 Algorithm NCSPunion

The graph X_k given by the algorithm NCSPsupergraph contains a shortest path for each terminal pair, but X_k may also contain edges that do not belong to any shortest path. To overcome this problem we apply algorithm NCSPunion, that builds a directed graph $Y_k = \bigcup_{i\in[k]} \rho_i$, where ρ_i is a directed shortest i-path, for $i \in [k]$. Moreover, we prove that Y_k can be built in linear time. This implies that, by using the results in [3], we can compute the length of all shortest i-paths, for $i \in [k]$, in $O(n)$ time (see Theorem 5).

We use the sequence of subgraphs $\{X_i\}_{i\in[k]}$. By Theorem 3, we know that X_i contains a shortest undirected i-path π_i and we can list its edges in $O(|\pi_i|)$ time. But if an edge e is shared by many π_i's, then e is visited many times. Thus obtaining $\bigcup_{i\in[k]} \pi_i$ by this easy procedure requires $O(kn)$ time. To overcome this problem, we should visit each edge in $\bigcup_{i\in[k]} \pi_i$ only a constant number of times.

Now we introduce two useful lemmata. The first lemma shows that two directed paths λ_i and λ_j that are uncomparable in the genealogy tree T_G (i.e., such that $i \not\preceq j$ and $j \not\preceq i$) cannot share a dart, although it is possible that $\overrightarrow{ab} \in \lambda_i$ and $\overrightarrow{ba} \in \lambda_j$. The second lemma deals with the intersection of non-crossing paths joining comparable pairs.

Lemma 2 *Let λ_i be a shortest directed i-path and let λ_j be a shortest directed j-path, for some $i, j \in [k]$. If j is not an ancestor neither a descendant of i in T_G, then λ_i and λ_j have no common darts.*

Proof: Let us assume by contradiction that λ_i and λ_j have at least one common dart, and let d be the dart in $\lambda_i \cap \lambda_j$ that appears first in λ_i. Let R be the region bounded by $\overline{\lambda_j[s_j, \text{tail}[d]]}$, $\overline{\lambda_i[s_i, \text{tail}[d]]}$ and the clockwise undirected s_i-s_j path in f^∞ (Figure 7(a) shows λ_i, λ_j and R). Being λ_j a simple path, then λ_j crosses λ_i in at least one vertex in $\lambda_i[s_i, \text{tail}[d]]$. Let x be the first vertex in $\lambda_i[s_i, \text{tail}[d]]$ after $\text{head}[d]$ in λ_j.

Now by looking at cycle $\lambda_i[x, \text{head}[d]] \circ \lambda_j[\text{head}[d], x]$ shown in Figure 7(b), we prove that λ_i and λ_j can be both shortest paths. Indeed, if λ_i is a shortest path, then $\lambda_j[s_j, \text{tail}[d]] \circ \lambda_i[\text{tail}[d], x] \circ \lambda_j[x, t_j]$ is shorter than λ_j because it does not contain d. Finally, if λ_j is a shortest path, then $\lambda_i[s_i, x] \circ \lambda_j[x, \text{head}[d]] \circ \lambda_i[\text{head}[d], t_i]$ is shorter than λ_i because it does not contain d. □

Lemma 3 *Let $\{\lambda_i\}_{i\in[k]}$ be a set of non-crossing directed i-paths. Let $i, j \in [k]$, if $i \preceq j$, then $\lambda_i \cap \lambda_j \subseteq \lambda_\ell$, for all $\ell \in [k]$ such that $i \preceq \ell \preceq j$.*

Proof: Let us assume λ_i and λ_j have at least one common vertex and choose $\ell \in [k]$ such that $i \preceq \ell \preceq j$. Let v be a vertex in $\lambda_i \cap \lambda_j$ and let Q be the region bounded by $\overline{\lambda_j[s_j, v]}$, $\overline{\lambda_i[s_i, v]}$ and the clockwise undirected s_j-s_i path in f^∞ (region Q and vertex v are shown in Figure 7(c)). It is clear that if $v \notin \lambda_\ell$, then $\{\lambda_i, \lambda_j, \lambda_\ell\}$ is not a set of non-crossing paths, absurdum. □

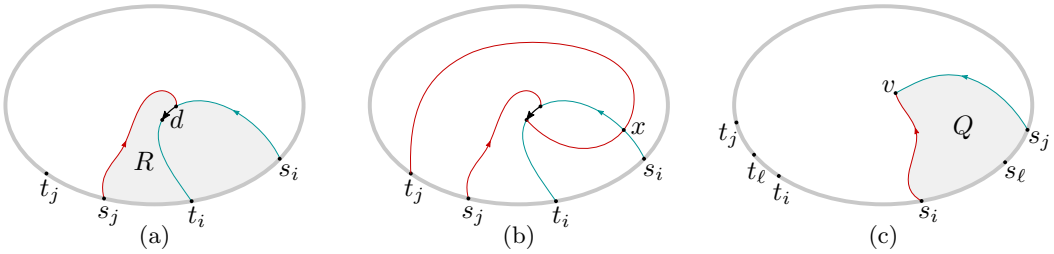


Figure 7: in (a) and (b) paths λ_j and λ_i, dart d, region R and vertex x used in the proof of Lemma 2. In (c) region Q and vertex v used in the proof of Lemma 3.

Now we show how to use these two lemmata for our goals. Let ρ_i be a shortest directed i-path and let ρ_j be a shortest directed j-path, for some $i, j \in [k]$, $i \neq j$. By Lemma 2, if i and j are not comparable in T_G, then ρ_i and ρ_j have no common darts. Moreover, by Lemma 3, if i is a descendant of j in T_G, then $\rho_i \cap \rho_j \subseteq \rho_{p(i)}$. By using these two facts, in order to list darts in ρ_i, then it suffices to find darts in $\rho_i \setminus \rho_{p(i)}$, for all $i \in [k] \setminus \{1\}$ (we remind that $i = 1$ is the root of T_G). To this goal we use algorithm `NCSPunion`, that builds a sequence of directed graphs $\{Y_i\}_{i\in[k]}$ such that Y_k is equal to $\bigcup_{i\in[k]} \rho_i$, where ρ_i is a shortest directed i-path, for $i \in [k]$.

We prove the correctness of algorithm `NCSPunion` in Theorem 4. At iteration i we compute $\rho_i \setminus \rho_{p(i)}$, showing that $\rho_i \setminus \rho_{p(i)} = \sigma_i \cup \mathrm{rev}[\tau_i]$, where σ_i and τ_i are computed in Line 5 and Line 6, respectively. We observe that if ρ_i and $\rho_{p(i)}$ have no common darts, then $\sigma_i = \mathrm{rev}[\tau_i] = \rho_i$.

To better understand Line 2 of algorithm `NCSPunion`, we recall that X_1 is an undirected 1-path, hence Y_1 is the directed version of this path.

Lemma 4 *Algorithm `NCSPunion` has $O(n)$ time complexity.*

Proof: Algorithm `NCSPunion` uses algorithm `NCSPsupergraph`, that has $O(n)$ time complexity by Lemma 1. Moreover, algorithm `NCSPunion` visits each dart of the "directed version" of X_k at most $O(1)$ times, where the *directed version of* X_k is the directed graph built from X_k by replacing each edge ab with the pair of darts $\overrightarrow{ab}$ and $\overrightarrow{ba}$. Thus, algorithm `NCSPunion` requires $O(n)$ time, since X_k is a subgraph of G. □

Theorem 4 *Graph Y_k computed by algorithm `NCSPunion` is the union of k shortest directed non-crossing i-paths, for $i \in [k]$.*

Proof: Let $\{\pi_i\}_{i\in[k]}$ be the set of paths defined in Theorem 3. For all $i \in [k]$, we denote by $\overrightarrow{\pi_i}$ the directed version of π_i, oriented from s_i to t_i.

First we define $\rho_1 = \overrightarrow{\pi_1}$ and for all $i \in [k] \setminus \{1\}$ we define

Algorithm `NCSPunion`**:**

Input: an undirected unweighted plane graph G and k well-formed terminal pairs $\{(s_i, t_i)\}_{i\in[k]}$ on the external face of G

Output: a directed graph Y_k formed by the union of directed non-crossing shortest paths from s_i to t_i, for $i \in [k]$

```
Compute X_1 as in algorithm NCSPsupergraph;
Y_1 is the directed version of X_1 oriented from s_1 to t_1;
for i = 2, ..., k do
    Compute X_i as in algorithm NCSPsupergraph;
    σ_i is the directed path that starts in s_i and always turns left in X_i until either σ_i
      reaches t_i or the next dart d_i of σ_i satisfies d_i ∈ Y_{i-1};
    τ_i is the directed path that starts in t_i and always turns right in X_i until either τ_i
      reaches s_i or the next dart d'_i of τ_i satisfies rev[d'_i] ∈ Y_{i-1};
    Y_i = Y_{i-1} ∪ σ_i ∪ rev[τ_i];
```

$$\rho_i = \begin{cases} \overrightarrow{\pi_i}[s_i, u_i] \circ \rho_{p(i)}[u_i, v_i] \circ \overrightarrow{\pi_i}[v_i, t_i], & \text{if } \overrightarrow{\pi_i} \text{ and } \rho_{p(i)} \text{ share at least one dart,} \\ \overrightarrow{\pi_i}, & \text{otherwise,} \end{cases} \tag{1}$$

where we assume that if $\overrightarrow{\pi_i}$ and $\rho_{p(i)}$ share at least one dart, then u_i and v_i are the tail of the first common dart and the head of the last common dart, respectively, where the order is with respect to $\overrightarrow{\pi_i}$. The definition of ρ_i as in (1) is shown in Figure 8. Now we split the proof into three parts: first we prove that $\{\rho_i\}_{i\in[k]}$ is a set of shortest paths (we need it to apply Lemma 2); second we prove that $\{\rho_i\}_{i\in[k]}$ is a set of non-crossing paths (we need it to apply Lemma 3); third we prove that $Y_k = \bigcup_{i\in[k]} \rho_i$ (we prove it by Lemma 2 and Lemma 3).

$\{\rho_i\}_{i\in[k]}$ **is a set of shortest paths:** we proceed by induction on i. The base case is trivial because π_1 is a shortest path by definition. Let us assume that ρ_j is a shortest j-path, for $j < i$, we have to prove that ρ_i is a shortest i-path. If $\overrightarrow{\pi_i}$ and $\rho_{p(i)}$ have no common darts, then $\rho_i = \overrightarrow{\pi_i}$ by (1), thus the thesis holds because $\{\pi_i\}_{i\in[k]}$ is a set of shortest paths. Hence let us assume that $\overrightarrow{\pi_i}$ and $\rho_{p(i)}$ have at least one common darts, then it suffices, by definition of ρ_i, that $|\pi_i[u_i, v_i]| = |\rho_{p(i)}[u_i, v_i]|$. It is true by induction.

$\{\rho_i\}_{i\in[k]}$ **is a set of non-crossing paths:** we proceed by induction on i. The base case is trivial because there is only one path. Let us assume that $\{\rho_j\}_{j\in[i-1]}$ is a set of non-crossing paths, we have to prove that ρ_i does not cross ρ_j, for any $j < i$.

By construction of ρ_i, ρ_i can not cross $\rho_{p(i)}$. Thus if ρ_i and ρ_j are crossing and j is not an ancestor of i, then either $\rho_{p(i)}$ and ρ_j are crossing or π_i and π_j are crossing; that is absurdum in both cases by induction and Theorem 3. Moreover if ℓ is an ancestor of i such that $\ell \neq p(i)$, then ρ_i does not cross ρ_ℓ otherwise ρ_ℓ would cross $\rho_{p(i)}$, absurdum by induction. Hence $\{\rho_i\}_{i\in[k]}$ is a set of non-crossing paths.

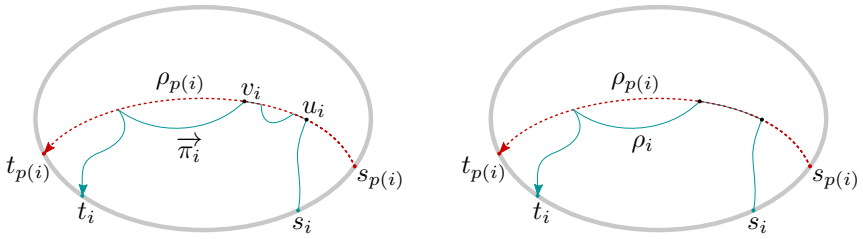
Y_k **is the union of** ρ_i**'s:** now we prove that $Y_k = \bigcup_{i\in[k]} \rho_i$. In particular we show that $\rho_1 = \overrightarrow{\pi_1}$ and for all $i \in [k] \setminus \{1\}$

$$\rho_i = \begin{cases} \sigma_i \circ \rho_{p(i)}[u_i, v_i] \circ \mathrm{rev}[\tau_i], & \text{if } \overrightarrow{\pi_i} \text{ and } \rho_{p(i)} \text{ share at least one dart,} \\ \overrightarrow{\pi_i}, & \text{otherwise.} \end{cases} \tag{2}$$

Again, we proceed by induction on i. The base case is trivial, thus we assume that (1) is equivalent to (2) for all $i < \ell$. We have to prove that (1) is equivalent to (2) for $i = \ell$.

If $\overrightarrow{\pi_\ell}$ does not intersect any dart of $\rho_{p(\ell)}$, then (1) is equivalent to (2). Thus we assume that $\overrightarrow{\pi_\ell}$ and $\rho_{p(\ell)}$ share at least one dart. By (1) and (2) and by definition of σ_i and τ_i in Line 5 and Line 6, respectively, it suffices to prove that $d_i \in \rho_{p(i)}$ and $\mathrm{rev}[d'_i] \in \rho_{p(i)}$.

Now, by induction we know that $d_i \in \rho_\ell$ for some $\ell < i$, we have to show that $d_i \in \rho_{p(i)}$. By Lemma 2 and being $\{\rho_j\}_{j\in[k]}$ a set of shortest paths, it holds that ℓ is an ancestor or a descendant of i. Being the s_j's visited clockwise by starting from s_1, then ℓ is an ancestor of i. Finally, by Lemma 3 and being $\{\rho_j\}_{j\in[k]}$ a set of non-crossing paths, it holds that $\rho_i \cap \rho_\ell \subseteq \rho_{p(i)}$. Being $p(i) < i$, then $d_i \in \rho_{p(i)}$ as we claimed. By a similar argument, it holds that $\mathrm{rev}[d'_i] \in \rho_{p(i)}$. □



(a) path $\overrightarrow{\pi_i}$ and $\rho_{p(i)}$ (b) path ρ_i

Figure 8: proof of Theorem 4, explanation of (1).

It is proved in [3] that, starting from the union of a set of shortest (not necessarily non-crossing) paths between well-formed terminal pairs, distances between terminal pairs can be computed in linear time. Thus we can give the following main theorem.

Theorem 5 *Given an undirected unweighted plane graph G and a set of well-formed terminal pairs $\{(s_i, t_i)\}$ on the external face f^∞ of G we can compute $U = \bigcup_{i\in[k]} p_i$ and the lengths of all p_i, for $i \in [k]$, where p_i is a shortest i-path and $\{p_i\}_{i\in[k]}$ is a set of non-crossing paths, in $O(n)$ time.*

Proof: By Theorem 4, the required graph U is the undirected version $\overline{Y_k}$ of the graph computed by algorithm `NCSPunion`, that has $O(n)$ time complexity by Lemma 4. Moreover, we compute the length of p_i, for all $i \in [k]$, in $O(n)$ time by using the results in [3]. □

Remark 2 *If G is a plane graph with small integer weights, then we can obtain all the previous results in $O(n+L)$ time, where L is the sum of all edge weights of G, by splitting an edge of weight r in G in r unweighted edges.*

5 Conclusions

In this paper we have shown a linear time algorithm to compute the union of non-crossing shortest paths whose extremal vertices are in the external face of an undirected unweighted plane graph.

The algorithm relies on the algorithm by Eisenstat and Klein for computing SSSP trees rooted on the vertices of the external face and on the novel concept of ISP subgraph of a plane graph,

that can be of interest itself. The same approach cannot be extended to weighted plane graphs, because the algorithm by Eisenstat and Klein works only in the unweighted case.

As stated in [16] our results may be applied in the case of terminal pairs lying on h face boundaries, where h is any positive integer.

We wish to investigate the non-crossing shortest paths problem when each terminal pair contains only one vertex on the external face.

Acknowledgements

The authors appreciate the unknown referee's valuable and deep comments.

References

[1] G. Ausiello, P. G. Franciosa, I. Lari, and A. Ribichini. Max-flow vitality in undirected unweighted planar graphs. *CoRR*, abs/2011.02375, 2020. URL: https://arxiv.org/abs/2011.02375, arXiv:2011.02375.

[2] Z. K. Baker and M. B. Gokhale. On the Acceleration of Shortest Path Calculations in Transportation Networks. In *IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2007*, pages 23–34, 2007. doi:10.1109/FCCM.2007.46.

[3] L. Balzotti and P. G. Franciosa. Computing Lengths of Non-Crossing Shortest Paths in Planar Graphs. *CoRR*, abs/2011.04047, 2020. URL: https://arxiv.org/abs/2011.04047, arXiv:2011.04047.

[4] L. Balzotti and P. G. Franciosa. Max Flow Vitality of Edges and Vertices in Undirected Planar Graphs. *CoRR*, abs/2201.13099, 2022. URL: https://arxiv.org/abs/2201.13099, arXiv:2201.13099.

[5] L. Balzotti and P. G. Franciosa. Non-Crossing Shortest Paths in Undirected Unweighted Planar Graphs in Linear Time. In *Computer Science - Theory and Applications - 17th International Computer Science Symposium in Russia, CSR 2022, Proceedings*, volume 13296 of *Lecture Notes in Computer Science*, pages 77–95. Springer, 2022. doi:10.1007/978-3-031-09574-0_6.

[6] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra's Algorithm. *ACM Journal of Experimental Algorithmics*, 15, 2010. doi:10.1145/1671970.1671976.

[7] M. Bentert, A. Nichterlein, M. Renken, and P. Zschoche. Using a Geometric Lens to Find k Disjoint Shortest Paths. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 26:1–26:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.26.

[8] K. Bérczi and Y. Kobayashi. The Directed Disjoint Shortest Paths Problem. In *25th Annual European Symposium on Algorithms, ESA 2017*, volume 87 of *LIPIcs*, pages 13:1–13:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.13.

[9] S. N. Bhatt and F. T. Leighton. A Framework for Solving VLSI Graph Layout Problems. *Journal of Computer and System Sciences*, 28(2):300–343, 1984. doi:10.1016/0022-0000(84)90071-0.

[10] S. Cabello. Many Distances in Planar Graphs. *Algorithmica*, 62(1-2):361–381, 2012. doi:10.1007/s00453-010-9459-0.

[11] D. Z. Chen and J. Xu. Shortest Path Queries in Planar Graphs. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 469–478. ACM, 2000. doi:10.1145/335305.335359.

[12] É. C. de Verdière and A. Schrijver. Shortest Vertex-Disjoint Two-Face Paths in Planar Graphs. *ACM Transactions on Algorithms*, 7(2):19:1–19:12, 2011. doi:10.1145/1921659.1921665.

[13] H. N. Djidjev. Efficient Algorithms for Shortest Path Queries in Planar Digraphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 151–165. Springer, 1996.

[14] T. Eilam-Tzoreff. The Disjoint Shortest Paths Problem. *Discrete Applied Mathematics*, 85(2):113–138, 1998. doi:10.1016/S0166-218X(97)00121-2.

[15] D. Eisenstat and P. N. Klein. Linear-Time Algorithms for Max Flow and Multiple-Source Shortest Paths in Unit-Weight Planar Graphs. In *Symposium on Theory of Computing Conference, STOC'13*, pages 735–744. ACM, 2013. doi:10.1145/2488608.2488702.

[16] J. Erickson and A. Nayyeri. Shortest Non-Crossing Walks in the Plane. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pages 297–208. SIAM, 2011. doi:10.1137/1.9781611973082.25.

[17] S. D. Eriksson-Bique, J. Hershberger, V. Polishchuk, B. Speckmann, S. Suri, T. Talvitie, K. Verbeek, and H. Yildiz. Geometric k Shortest Paths. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1616–1625. SIAM, 2015. doi:10.1137/1.9781611973730.107.

[18] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72(5):868–889, 2006. doi:10.1016/j.jcss.2005.05.007.

[19] G. N. Frederickson. Fast Algorithms for Shortest Paths in Planar Graphs, with Applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987. doi:10.1137/0216064.

[20] P. Gawrychowski, S. Mozes, O. Weimann, and C. Wulff-Nilsen. Better Tradeoffs for Exact Distance Oracles in Planar Graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 515–529. SIAM, 2018. doi:10.1137/1.9781611975031.34.

[21] A. V. Goldberg. Point-to-Point Shortest Path Algorithms with Preprocessing. In *SOFSEM 2007: 33rd Conference on Current Trends in Theory and Practice of Computer Science*, volume 4362 of *Lecture Notes in Computer Science*, pages 88–102. Springer, 2007. doi:10.1007/978-3-540-69507-3_6.

[22] J. L. Gross and T. W. Tucker. *Topological Graph Theory*. Courier Corporation, 2001.

[23] R. Hassin. Maximum Flow in (s,t) Planar Networks. *Information Processing Letters*, 13(3):107, 1981. doi:10.1016/0020-0190(81)90120-4.

[24] R. Hassin and D. B. Johnson. An $O(n \log^2 n)$ Algorithm for Maximum Flow in Undirected Planar Networks. *SIAM Journal on Computing*, 14(3):612–624, 1985. doi:10.1137/0214045.

[25] M. R. Henzinger, P. N. Klein, S. Rao, and S. Subramanian. Faster Shortest-Path Algorithms for Planar Graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997. doi:10.1006/jcss.1997.1493.

[26] G. F. Italiano, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen. Improved Algorithms for Min Cut and Max Flow in Undirected Planar Graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 313–322. ACM, 2011. doi:10.1145/1993636.1993679.

[27] N. Jing, Y. Huang, and E. A. Rundensteiner. Hierarchical Optimization of Optimal Path Finding for Transportation Applications. In *CIKM '96, Proceedings of the Fifth International Conference on Information and Knowledge Management*, pages 261–268. ACM, 1996. doi:10.1145/238355.238550.

[28] D. Kim and N. F. Maxemchuk. Simple Robotic Routing in Ad Hoc Networks. In *13th IEEE International Conference on Network Protocols (ICNP 2005)*, pages 159–168. IEEE Computer Society, 2005. doi:10.1109/ICNP.2005.37.

[29] P. N. Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 146–155. SIAM, 2005. URL: http://dl.acm.org/citation.cfm?id=1070432.1070454.

[30] Y. Kobayashi and C. Sommer. On shortest disjoint paths in planar graphs. *Discrete Optimization*, 7(4):234–245, 2010. doi:10.1016/j.disopt.2010.05.002.

[31] L. Kowalik and M. Kurowski. Short Path Queries in Planar Graphs in Constant Time. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 143–148. ACM, 2003. doi:10.1145/780542.780565.

[32] Y. Kusakari, D. Masubuchi, and T. Nishizeki. Finding a Noncrossing Steiner Forest in Plane Graphs Under a 2-Face Condition. *Journal of Combinatorial Optimization*, 5(2):249–266, 2001. doi:10.1023/A:1011425821069.

[33] F. T. Leighton. *Complexity issues in VLSI: optimal layouts for the shuffle-exchange graph and other networks*. MIT press, 1983.

[34] F. T. Leighton. New Lower Bound Techniques for VLSI. *Mathematical systems theory*, 17(1):47–70, 1984.

[35] W. Lochet. A Polynomial Time Algorithm for the k-Disjoint Shortest Paths Problem. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 169–178. SIAM, 2021. doi:10.1137/1.9781611976465.12.

[36] A. P. Masucci, K. Stanilov, and M. Batty. Exploring the evolution of London's street network in the information space: A dual approach. *Physical Review E*, 89(1):012805, 2014.

[37] S. Mozes and C. Sommer. Exact Distance Oracles for Planar Graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 209–222. SIAM, 2012. doi:10.1137/1.9781611973099.19.

[38] Y. Nussbaum. Improved Distance Queries in Planar Graphs. In *Algorithms and Data Structures - 12th International Symposium, WADS*, volume 6844 of *Lecture Notes in Computer Science*, pages 642–653. Springer, 2011. doi:10.1007/978-3-642-22300-6_54.

[39] E. Papadopoulou. k-Pairs Non-Crossing Shortest Paths in a Simple Polygon. *International Journal of Computational Geometry and Applications*, 9(6):533–552, 1999. doi:10.1142/S0218195999000315.

[40] B. Raney and K. Nagel. Iterative route planning for large-scale modular transportation simulations. *Future Generation Computer Systems*, 20(7):1101–1118, 2004. doi:10.1016/j.future.2003.11.001.

[41] J. H. Reif. Minimum s-t Cut of a Planar Undirected Network in $O(n \log^2(n))$ time. *SIAM Journal on Computing*, 12(1):71–81, 1983. doi:10.1137/0212005.

[42] A. Schrijver. Finding k Disjoint Paths in a Directed Planar Graph. *SIAM Journal on Computing*, 23(4):780–788, 1994. doi:10.1137/S0097539792224061.

[43] A. J. Steiger. Single-Face Non-Crossing Shortest Paths in Planar Graphs. *M.S. thesis, University of Illinois at Urbana-Champaign*, 2017. URL: https://hdl.handle.net/2142/98345.

[44] J. Takahashi, H. Suzuki, and T. Nishizeki. Shortest Noncrossing Paths in Plane Graphs. *Algorithmica*, 16(3):339–357, 1996. doi:10.1007/BF01955681.

[45] J. Takahashi, H. Suzuki, and T. Nishizeki. Shortest Non-Crossing Rectilinear Paths in Plane Regions. *International Symposium on Algorithms and Computation*, 7(5):419–436, 1997. doi:10.1142/S0218195997000259.

[46] D. Wagner and K. Weihe. A Linear-Time Algorithm for Edge-Disjoint Paths in Planar Graphs. *Combinatorica*, 15(1):135–150, 1995. doi:10.1007/BF01294465.

[47] A. Ziliaskopoulos, D. Kotzinos, and H. S. Mahmassani. Design and implementation of parallel time-dependent least time path algorithms for intelligent transportation systems applications. *Transportation Research Part C: Emerging Technologies*, 5(2):95–107, 1997.