

Alexander S. Kulikov
Sofya Raskhodnikova (Eds.)

LNCS 13296

Computer Science – Theory and Applications

17th International Computer Science Symposium in Russia, CSR 2022
Virtual Event, June 29 – July 1, 2022
Proceedings



 Springer

Founding Editors

Gerhard Goos

Karlsruhe Institute of Technology, Karlsruhe, Germany

Juris Hartmanis

Cornell University, Ithaca, NY, USA


Editorial Board Members

Elisa Bertino

Purdue University, West Lafayette, IN, USA

Wen Gao

Peking University, Beijing, China

Bernhard Steffen 

TU Dortmund University, Dortmund, Germany

Moti Yung 

Columbia University, New York, NY, USA


More information about this series at <https://link.springer.com/bookseries/558>

Alexander S. Kulikov ·
Sofya Raskhodnikova (Eds.)

Computer Science – Theory and Applications

17th International Computer Science Symposium in Russia, CSR 2022
Virtual Event, June 29 – July 1, 2022
Proceedings

Editors

Alexander S. Kulikov 
Steklov Institute of Mathematics
St. Petersburg, Russia

Sofya Raskhodnikova 
Boston University
Boston, MA, USA

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-3-031-09573-3

ISBN 978-3-031-09574-0 (eBook)

<https://doi.org/10.1007/978-3-031-09574-0>

© Springer Nature Switzerland AG 2022

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the papers presented at CSR 2022, the 17th International Computer Science Symposium in Russia, held online during June 29 – July 1, 2022. CSR covers a wide range of areas in theoretical computer science and its applications. Initially, CSR 2022 was planned as a satellite event for the International Congress of Mathematicians (ICM) in St. Petersburg, Russia. However, as the Program Committee (PC) was starting its deliberations after completing submission reviews, Russia attacked Ukraine. As a result, ICM and CSR 2022 were moved online. Many PC members expressed dismay at the attack and three PC members resigned. Others chose to continue their work, but many wanted to emphasize that they did not support or condone the actions of the Russian government against Ukrainian people.

We received 51 submissions, and out of these the Program Committee selected 21 papers for presentation at the symposium and for publication in the proceedings. Each submission was reviewed by at least three Program Committee members. Submissions by Program Committee members were reviewed by at least four other members of the Program Committee.

The opening lecture at CSR 2022 was given by Umesh Vazirani (University of California at Berkeley), the closing lecture was given by Mark Braverman (Princeton University). Three invited plenary lectures were given by Irit Dinur (Weizmann Institute of Science), Jelani Nelson (University of California at Berkeley), and Mary Wootters (Stanford University).

Many people and organizations contributed to the smooth running and the success of CSR 2022. In particular, our thanks go to



- all authors who submitted their work to CSR;
- the members of the Program Committee who graciously devoted their time and energy to the evaluation process;
- the expert reviewers who helped us evaluate the papers;
- the invited speakers; and
- the members of the local Organizing Committee who made the conference possible.

May 2022

Alexander S. Kulikov
Organizing Committee Chair
Sofya Raskhodnikova
Program Committee Chair



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

Lorenzo Balzotti¹ and Paolo G. Franciosa²

¹ Dipartimento di Scienze di Base e Applicate per l'Ingegneria,
Sapienza Università di Roma, Via Antonio Scarpa, 16, 00161 Rome, Italy
`lorenzo.balzotti@sbai.uniroma1.it`

² Dipartimento di Scienze Statistiche, Sapienza Università di Roma,
p.le Aldo Moro 5, 00185 Rome, Italy
`paolo.franciosa@uniroma1.it`

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.

Keywords: planar graphs · non-crossing paths · shortest paths · undirected unweighted graphs · multiple pairs · external face

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) [22, 36], route planning [6, 16, 30], logistic [27], traffic simulations [3] and robotics [23]. In particular, non-crossing paths in a planar graph are studied to optimize VLSI layout [7], where two *non-crossing* paths may share edges and vertices, but they do not cross each other in the plane.

We are given a planar graph $G = (V, E)$, where V is a set of n vertices and E is a set of edges, with $|E| = O(n)$. The graph has a fixed embedding, and we are also given a set of k terminal pairs $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$ 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 Subsect. 2.2).

State of the Art. Takahashi *et al.* [33] solved the NCSP problem in a non-negative edge-weighted planar graph in $O(n \log k)$ time (actually, in their paper the time complexity is $O(n \log n)$, that can easily be reduced to $O(n \log k)$ by applying the planar single source shortest path algorithm by Henzinger *et al.* [20]). Their result is improved by Steiger in $O(n \log \log k)$ time [32], exploiting the algorithm by Italiano *et al.* [21]. These two algorithms maintain the same time complexity also in the unweighted case.

Our Results. In this paper, we solve the NCSP problem on unweighted planar graphs in $O(n)$ time. We improve, in the unweighted case, the results in [32, 33]. By applying the technique in [4] we can compute distances between all terminal pairs in linear time.

Our algorithm relies on two main results:

- an algorithm due to Eisenstat and Klein [11], that gives in $O(n)$ time an implicit representation of a sequence of shortest-path trees in an undirected unweighted planar graph G , where each tree is rooted in 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 [11], 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 graph G , introduced in Sect. 3, that is a subgraph incrementally built by adding a sequence of shortest paths in G starting from the infinite face of 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 .

Related Work. Our article 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 [14], while the single source shortest paths (SSSP) problem is solved in linear time by Henzinger *et al.* [20]. The best known algorithm for computing many distances in planar graphs is due to Gawrychowski *et al.* [15] 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 planar unweighted case, SSSP trees rooted at vertices in the external face can be computed in linear time as in [11]. More results on many distances problem can be found in [8–10, 13, 28, 29].

If we are interested in distances from any vertex in the external face to any other vertex, then we can use Klein’s algorithm [24] that, with a preprocessing of $O(n \log n)$ time, answers to each distance query in $O(\log n)$ time.

Kowalik and Kurowski [25] 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 [18, 19, 31]. In particular, they are used to compute the vitality of edges and vertices with respect to the max-flow [1, 2, 5].

Balzotti and Franciosa [4] show that, given the union of a set of non-crossing shortest paths in a planar graph, the lengths of each shortest path can be computed in linear time. This improves the result of [33], that can only be applied when the union of the shortest paths is a forest.

Wagner and Weihe [35] present an $O(n)$ time algorithm for finding edge-disjoint (not necessarily shortest) paths in a undirected planar graph such that each path connects two specified vertices on the infinite face of the graph.

Improved Results. We specialize the problem of finding k non-crossing shortest paths in [33] 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 [12, 26, 34].

Erickson and Nayyeri [12] generalized the work in [33] 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 [33] used their algorithm to compute k non-crossing rectilinear paths with minimum total length in a plane with r obstacles [34]. 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.* [26] showed that a set of non-crossing forests in a planar graph can be found in $O(n \log n)$ time, where two forest 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*. We introduce a new class of graphs, ISP subgraphs, that partition a planar graph into regions that preserve distances. Our algorithm is split in two parts.

In the first part we use Eisenstat and Klein's algorithm that gives a sequence of shortest path trees rooted in the vertices of the external face. We choose some specific shortest paths from each tree to obtain a sequence of ISP subgraphs X_1, \dots, 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 $i \in \{1, \dots, k\}$.

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 Sect. 2, in Sect. 3 we explain the main theoretical novelty. In Sect. 4 first we resume Eisenstat and Klein's algorithm in Subsect. 4.1, then in Subsects. 4.2 and 4.3 we show the two parts of our algorithm, and we prove the whole computational complexity. Conclusions are given in Sect. 5.

2 Definitions

Let G be a plane graph, i.e., a planar graph with a fixed planar embedding. We denote by f_G^∞ (or simply f^∞) its unique infinite 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 [17].

We recall standard union and intersection operators on graphs, for convenience we define the empty graph as a graph without edges.

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:*

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

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 \vec{uv} the dart from u to v . For every dart \vec{uv} we define $\text{rev}[\vec{uv}] = \vec{vu}$ and $\text{head}[\vec{uv}] = v$. For every vertex $v \in V(G)$ we define the *degree* of v as $\text{deg}(v) = |\{e \in E(G) \mid v \text{ is an endpoint of } e\}|$.

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

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 \bar{p} its undirected version, in which each dart \vec{ab} is replaced by edge ab ; moreover, we denote by $\text{rev}[p]$ its reverse version, in which each dart \vec{ab} is replaced by dart \vec{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 $w(p)$ 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.

We say that two paths in a plane graph G 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 or darts. 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* [17]. Crossing and non-crossing paths are given in Fig. 1.

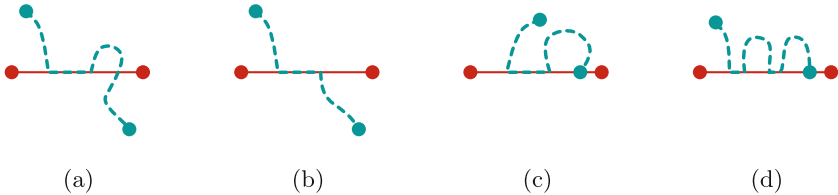


Fig. 1. paths in (a) and (b) are crossing, while paths in (c) and (d) are non-crossing.

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_G^∞ 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 $\gamma_j \subset \gamma_\ell$ or $\gamma_j \supset \gamma_\ell$ 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 [4,33] 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 . We say that $i \prec j$ if $\gamma_i \subset \gamma_j$. We define the *genealogy tree* T_G of a set of well-formed terminal pairs as the transitive reduction of poset $([k], \prec)$. W.l.o.g., we assume that $i^* = 1$, hence the root of T_G is 1.

If $i \prec j$, 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 $i \prec j$ and there is no r such that $i \prec r$ and $r \prec j$. 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_G^∞ by a solid light grey line. W.l.o.g., we assume that the external face is a simple cycle, hence, G is a biconnected graph. Indeed, if not, it suffices to solve the NCSP problem in each biconnected component.

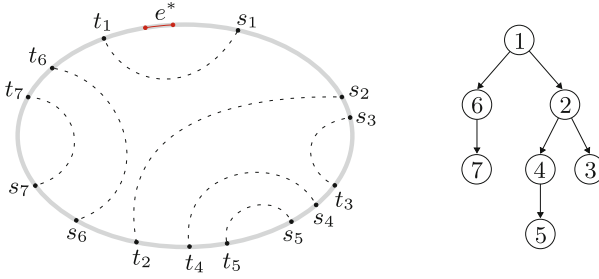


Fig. 2. on the left a set of well-formed terminal pairs. Any value in $\{1, 3, 5, 7\}$ can be chosen as i^* . If we choose $i^* = 1$, then we obtain the genealogy tree on the right.

3 ISP Subgraphs

In this section we introduce the concept of *incremental shortest paths (ISP) subgraph* of a graph G , that is a subgraph incrementally built by adding a sequence of shortest paths in G starting from f_G^∞ (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 [11].

Definition 2. A graph X is an incremental shortest paths (ISP) subgraph of a positive weighted graph G if $X = X_r$, where X_0, X_1, \dots, X_r is a sequence of subgraphs of G built in the following way: $X_0 = f_G^\infty$ 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_G^∞ .

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 Fig. 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 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 $(v_1, v_2, \dots, v_{2r})$ 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 every $i \in [r]$,

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 $\pi \downarrow C = \pi[a, v_1] \circ \mu_1 \circ \pi[v_2, v_3] \circ \mu_2 \dots \circ \pi[v_{2r-2}, v_{2r-1}] \circ \mu_r \circ \pi[v_{2r}, b]$.

Definition 2 and Definition 3 are depicted in Fig. 3.

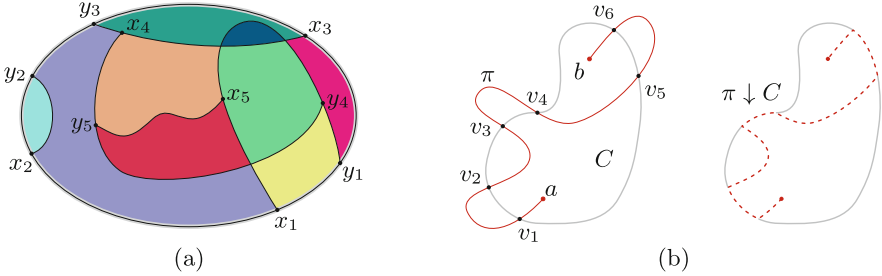


Fig. 3. (a) an ISP subgraph X of G ; extremal vertices x_i, y_i of p_i are drawn, for $i \in [5]$. Different faces of X have different colors. An example of Definition 3 is given in (b).

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 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 $w(\pi \downarrow \partial f) \leq w(\pi)$.*

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 j for every choice of a face f of X_j , $a, b \in R_{\partial f}$ and a - b path π .

In the base case, where $j = 1$, there are exactly two faces A and B in X_1 other than f_G^∞ . Let $a, b \in V(R_{\partial A})$ (the same argument holds for B) and let π be any a - b path. In case $\pi \subseteq R_{\partial A}$ we have $\pi \downarrow \partial A = \pi$, hence the thesis trivially holds. In case $\pi \not\subseteq R_{\partial A}$, then $\pi \downarrow \partial A$ is not longer than π because some subpaths of π have been replaced by subpaths of p_1 with the same extremal vertices and p_1 is a shortest path.

We assume that the thesis holds for all $i < j$ and we prove it for j . Let f be a face of X_j and let f' be the unique face of X_{j-1} such that $f \subset f'$ (Fig. 4(a) and Fig. 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_j 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_j with the same extremal vertices and p_j is a shortest path (see Fig. 4(c) where π is the red and 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 $w(\pi') \leq w(\pi)$. We observe now that $\pi' \subseteq R_{\partial f'}$ and $\pi' \not\subseteq R_{\partial f}$, hence the previous case applies, showing that $w(\pi' \downarrow \partial f) \leq w(\pi')$. Finally, the two previous inequalities imply $w(\pi \downarrow \partial f) \leq w(\pi \downarrow \partial f') \leq w(\pi)$ (see Fig. 4(c) where π is the green and continue path). \square

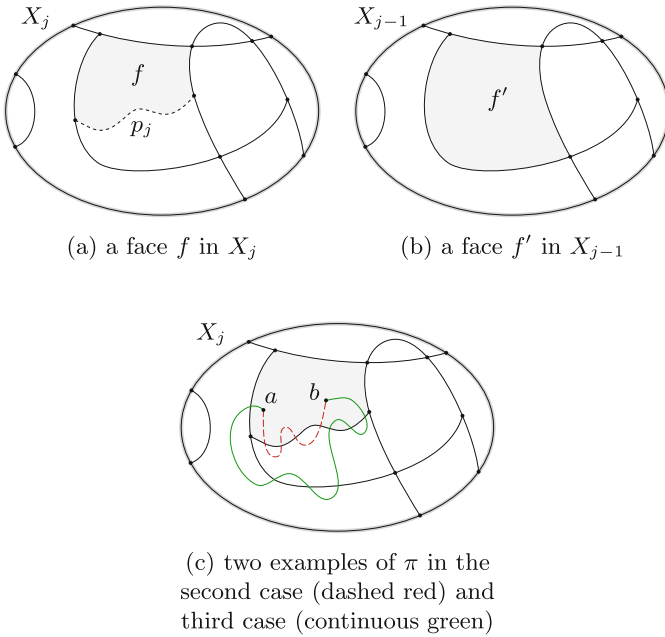


Fig. 4. in (a) and (b) faces f and f' build on the ISP graph in Fig. 3(a). In (c) we depict the second and third case of the proof of Theorem 1.

We can state now the main property of ISP subgraphs.

Corollary 1. *Let X be an ISP subgraph of G and let f be any 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 Subsect. 4.1 the result of Eisenstat and Klein’s paper [11], that deals with the multiple-source shortest paths problem. For the sake of clarity, we split our algorithm in two parts:

- in Subsect. 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 $X_i \cup f_G^\infty$ is an ISP subgraph of G , for all $i \in [k]$.
- in Subsect. 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 [11] takes as input an undirected unweighted planar graph G , where v_1, v_2, \dots, 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}_i , for $i \in [r]$, where each \mathcal{T}_i is rooted in v_i .

The sequence of trees \mathcal{T}_i , for $i \in [r]$, is represented by explicitly listing the darts in \mathcal{T}_1 , and listing the darts that are added to transform \mathcal{T}_i into \mathcal{T}_{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}_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}_1 and a sequence of sets of darts. A key result in [11] shows that if a dart d appears in $\mathcal{T}_{i+1} \setminus \mathcal{T}_i$, then d cannot appear in any $\mathcal{T}_{j+1} \setminus \mathcal{T}_j$, 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_G^∞ , but we only need the shortest path trees rooted in s_i ’s. So, we define T_i as the shortest path tree rooted in 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 [11] are rooted directed tree, thus $\overline{T_1[t_1]}$ is the undirected version of T_1). Then the sequence of subgraphs X_i , for $i = 2, \dots, k$ 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 passing from T_{i-1} to 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 .

Algorithm NCSPsupergraph:

Input: an undirected unweighted planar embedded graph G and k well-formed terminal pairs of vertices (s_i, t_i) , for $i \in [k]$, on the external face of G

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

- 1 Compute a shortest path tree T_1 rooted in s_1 ;
 - 2 $X_1 = \overline{T_1[t_1]}$;
 - 3 **for** $i = 2, \dots, k$ **do**
 - 4 $X_i = X_{i-1}$;
 - 5 Compute T_i from T_{i-1} by the algorithm of Eisenstat and Klein [11];
 - 6 Compute the set H_i of vertices of X_i whose parent in T_i differs from the parent in T_{i-1} ;
 - 7 For all $h \in H_i$, $X_i = X_i \cup \overline{T_i[h]}$;
 - 8 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 $X_i = X_i \cup \eta_i$;
-

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

Subgraphs $\{X_i\}_{i \in [k]}$ built by algorithm NCSPsupergraph, together with f_G^∞ , 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 $P = \{\pi_1, \dots, \pi_k\}$, where π_i is a shortest i -path, for $i \in [k]$. The main idea is to show that X_i contains an

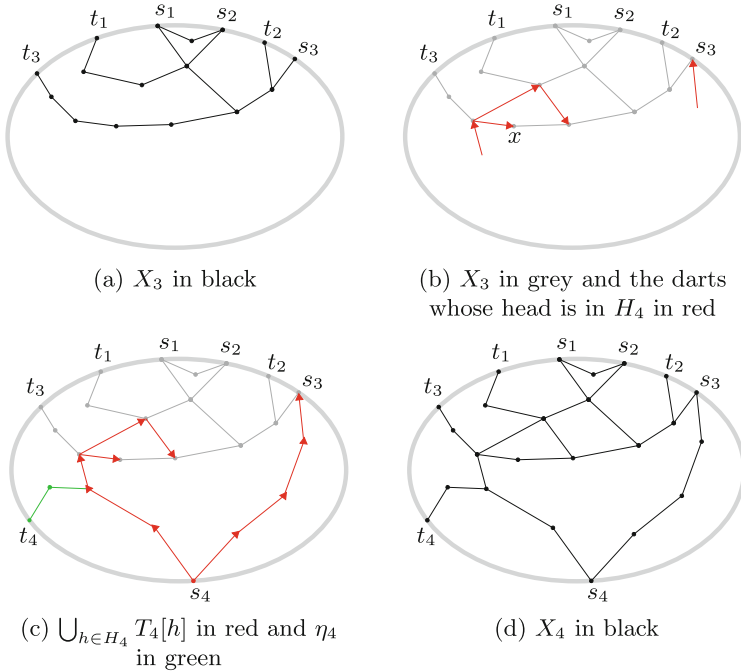


Fig. 5. algorithm NCSPSupergraph: graph X_4 is built starting from X_3 .

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 2.

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 not the leftmost a - b path is not necessarily the path that starts in a and always turns left until b is reached.

Theorem 2. *Let π_i be the undirected leftmost i -path in X_i , for $i \in [k]$. The following statements hold:*

- 2.(1) π_i is the s_i - t_i path in X_i that always turns left, for $i \in [k]$,
- 2.(2) π_i is a shortest i -path, for $i \in [k]$,
- 2.(3) for all $i, j \in [k]$, π_i and π_j are non-crossing.

Proof. We prove all the statements separately.

2.(1) For convenience, for every $i \in [k]$, 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 and, by Remark 1, $x \in f_{\infty}^{\circ}$. 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 . Assume by contradiction that $x \neq t_i$. Two cases are possible: either $x \in V(f_G^\infty) \setminus V(\gamma_i)$ 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 $\text{head}[d] = x$. 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 $\lambda_i = \pi_i$.

2.(2) We prove that π_i is a shortest i -path by using Theorem 1, indeed, $X_i \cup f_G^\infty$ 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_G^∞ 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 .

2.(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. □

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 [4], we can compute the length of all shortest i -paths, for $i \in [k]$, in $O(n)$ time (see Theorem 4).

We use the sequence of subgraphs $\{X_i\}_{i \in [k]}$. By Theorem 2, 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 every edge in $\bigcup_{i \in [k]} \pi_i$ only a constant number of times.

Now we introduce two useful lemmata that will be used later. The first lemma shows that two uncomparable directed paths π_i and π_j (i.e., such that $i \not\prec j$ and $j \not\prec i$) in the genealogy tree T_G cannot share a dart, although it is possible that $\overrightarrow{ab} \in \pi_i$ and $\overrightarrow{ba} \in \pi_j$. The second lemma deals with the intersection of non-crossing paths joining comparable pairs.

Lemma 2. *Let π_i be a shortest directed i -path and let π_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 π_i and π_j have no common darts.*

Proof. Let us assume by contradiction that π_i and π_j have at least one common dart, and let d be the dart in $\pi_i \cap \pi_j$ that appears first in π_i . Let R be the region bounded by $\overline{\pi_j[s_j, \text{tail}(d)]}$, $\overline{\pi_i[s_i, \text{tail}(d)]}$ and the clockwise undirected $s_i - s_j$ path in f^∞ (Fig. 6(a) shows π_i , π_j and R). Being π_j a simple path, then π_j crosses π_i in at least one vertex in $\pi_i[s_i, \text{tail}(d)]$. Let x be the first vertex in $\pi_i[s_i, \text{tail}(d)]$ after $\text{head}(d)$ in π_j . Now by looking to the cycle $\pi_i[x, \text{head}(d)] \circ \pi_j[\text{head}(d), x]$, it follows that π_i and π_j can be both shortest paths, absurdum (Fig. 6(b) shows this cycle). \square

Lemma 3. *Let $\{\pi_i\}_{i \in [k]}$ be a set of non-crossing directed paths. Let $i, j \in [k]$, if i is a descendant of j , then $\pi_i \cap \pi_j \subseteq \pi_\ell$, for all $\ell \in [k]$ such that $i \prec \ell \prec j$.*

Proof. Let us assume π_i and π_j have at least one common vertex and choose $\ell \in [k]$ such that $i \prec \ell \prec j$. Let v be a vertex in $\pi_i \cap \pi_j$ and let Q be the region bounded by $\overline{\pi_j[s_j, v]}$, $\overline{\pi_i[s_i, v]}$ and the clockwise undirected $s_j - s_i$ path in f^∞ (region Q and vertex v are shown in Fig. 6(c)). It is clear that if $v \notin \pi_\ell$, then $\{\pi_i, \pi_j, \pi_\ell\}$ is not a set of non-crossing paths, absurdum. \square

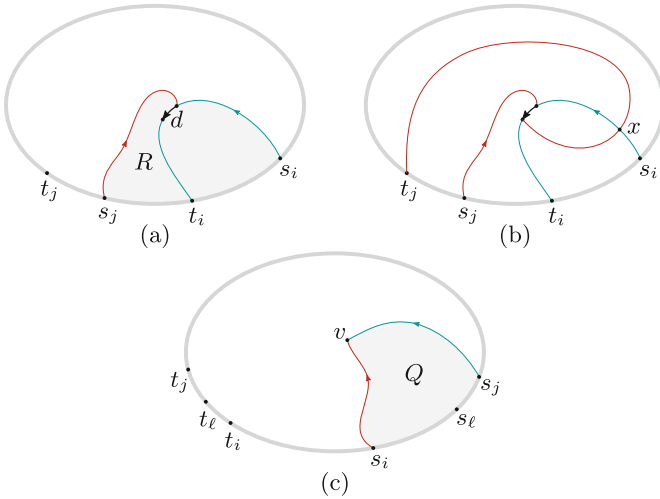


Fig. 6. in (a) and (b) paths π_j and π_i , dart d , region R and vertex x used in the proof of Lemma 2. In (c) region Q and vertex v used in the proof of Lemma 3.

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 an ancestor of j in T_G , then $\rho_i \cap \rho_j \subseteq \rho_{p(j)}$. 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 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 3. At iteration i we compute $\rho_i \setminus \rho_{p(i)}$, showing that $\rho_i \setminus \rho_{p(i)} = \sigma_i \cup \text{rev}[\tau_i]$, 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 $\sigma_i = \text{rev}[\tau_i] = \rho_i$.

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.

Algorithm NCSPunion:

Input: an undirected unweighted planar embedded graph G and k well-formed terminal pairs of vertices (s_i, t_i) , for $i \in [k]$, 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 $i \in [k]$

- 1 Compute X_1 as in algorithm **NCSPsupergraph**;
 - 2 Y_1 is the directed version of X_1 oriented from s_1 to t_1 ;
 - 3 **for** $i = 2, \dots, k$ **do**
 - 4 Compute X_i as in algorithm **NCSPsupergraph**;
 - 5 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 $d_i \in Y_{i-1}$;
 - 6 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 $\text{rev}[d'_i] \in Y_{i-1}$;
 - 7 $Y_i = Y_{i-1} \cup \sigma_i \cup \text{rev}[\tau_i]$;
-

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 by the pair of darts \overrightarrow{ab} and \overrightarrow{ba} . Thus, algorithm **NCSPunion** requires $O(n)$ time, since X_k is a subgraph of G . \square

Theorem 3. *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 2. 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

$$\rho_i = \begin{cases} \overrightarrow{\pi}_i[s_i, u_i] \circ \rho_{p(i)}[u_i, v_i] \circ \overrightarrow{\pi}_i[v_i, t_i], & \text{if } \overrightarrow{\pi}_i \text{ and } \rho_{p(i)} \text{ share no darts,} \\ \overrightarrow{\pi}_i, & \text{otherwise,} \end{cases} \quad (1)$$

where we assume that if $V(\overrightarrow{\pi}_i \cap \rho_{p(i)}) \neq \emptyset$, then u_i and v_i are the vertices in $V(\overrightarrow{\pi}_i \cap \rho_{p(i)})$ that appear first and last in $\overrightarrow{\pi}_i$, respectively; the definition of ρ_i as in (1) is shown in Fig. 7. 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 $Y = \bigcup_{i \in [k]} \rho_i$ (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 $\{\pi_i\}_{i \in [k]}$ a set of shortest paths. Hence let us assume that $\overrightarrow{\pi}_i$ and $\rho_{p(i)}$ have at least one common dart, 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 $j < i$.

If ρ_i and ρ_j are crossing and j is not an ancestor of i , then, by construction of ρ_i , 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 2. Moreover, by definition, ρ_i does not cross $\rho_{p(i)}$, and by induction, if ℓ is an ancestor of i such that $\ell \neq p(i)$, then ρ_i does not cross ρ_ℓ , indeed, if not, then ρ_ℓ would cross $\rho_{p(i)}$, absurdum. Hence $\{\rho_i\}_{i \in [k]}$ is a set of non-crossing paths.

Y is the union of ρ_i 's: now we prove that $Y = \bigcup_{i \in [k]} \rho_i$. In particular we show that $\rho_1 = \overrightarrow{\pi}_1$ and for all $i \in [k] \setminus \{1\}$

$$\rho_i = \begin{cases} \sigma_i \circ \rho_{p(i)}[u_i, v_i] \circ \text{rev}[\tau_i], & \text{if } \overrightarrow{\pi}_i \text{ and } \rho_{p(i)} \text{ share no darts,} \\ \overrightarrow{\pi}_i, & \text{otherwise.} \end{cases} \quad (2)$$

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 $\rho_{p(\ell)}$ have at least one common 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 $\text{rev}[d'_i] \in \rho_{p(i)}$.

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 path, 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 $\text{rev}[d'_i] \in \rho_{p(i)}$. \square

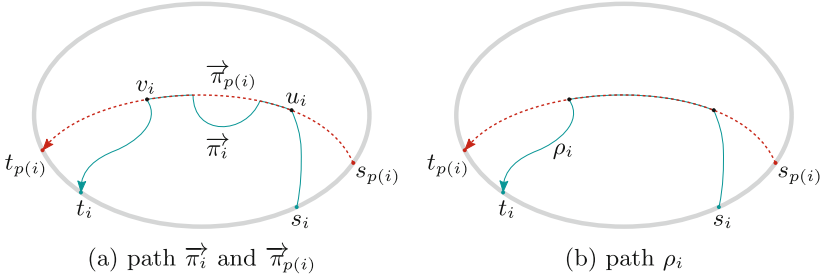


Fig. 7. Proof of Theorem 3, explanation of (1).

It is proved in [4] 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 4. *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 3, the required graph U is the undirected version $\overline{Y_k}$ of the graph computed by algorithm `NCSUnion`, 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 [4]. \square

Remark 2. For graphs with small integer weights, we can obtain all the previous results in $O(n + L)$ time, where L is the sum of all edge weights, by splitting an edge of weight r 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 planar 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 planar graph, that can be of interest itself. The same approach cannot be extended to weighted graphs, because the algorithm of Eisenstat and Klein works only in the unweighted case.

As stated in [12] our results may be applied in the case of terminal pairs lying on h face boundaries.

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

References

1. Ausiello, G., Balzotti, L., Franciosa, P.G., Lari, I., Ribichini, A.: A Linear Time Algorithm for Computing Max-Flow Vitality in Undirected Unweighted Planar Graphs, *CoRR*, abs/2204.10568 (2022)
2. Ausiello, G., Franciosa, P.G., Lari, I., Ribichini, A.: Max flow vitality in general and ST-planar graphs. *Networks* **74**, 70–78 (2019)
3. Baker, Z.K., Gokhale, M.B.: On the acceleration of shortest path calculations in transportation networks. In: *IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2007*, pp. 23–34 (2007)
4. Balzotti, L., Franciosa, P.G.: Computing Lengths of Non-Crossing Shortest Paths in Planar Graphs, *CoRR*, abs/2011.04047 (2020)
5. Balzotti, L., Franciosa, P.G.: Max Flow Vitality of Edges and Vertices in Undirected Planar Graphs, *CoRR*, abs/2201.13099 (2022)
6. Bauer, R., Dellinger, D., Sanders, P., Schieferdecker, D., Schultes, D., Wagner, D.: Combining hierarchical and goal-directed speed-up techniques for Dijkstra’s algorithm. *ACM J. Exp. Algorithmics* **15** (2010)
7. Bhatt, S.N., Leighton, F.T.: A framework for solving VLSI graph layout problems. *J. Comput. Syst. Sci.* **28**, 300–343 (1984)
8. Cabello, S.: Many distances in planar graphs. *Algorithmica* **62**, 361–381 (2012)
9. Chen, D.Z., Xu, J.: Shortest path queries in planar graphs. In: *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pp. 469–478. ACM (2000)
10. Djidjev, H.N.: Efficient algorithms for shortest path queries in planar digraphs. In: d’Amore, F., Franciosa, P.G., Marchetti-Spaccamela, A. (eds.) *WG 1996*. LNCS, vol. 1197, pp. 151–165. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62559-3_14
11. Eisenstat, D., Klein, P.N.: Linear-time algorithms for max flow and multiple-source shortest paths in unit-weight planar graphs. In: *Symposium on Theory of Computing Conference*, pp. 735–744. ACM (2013)
12. Erickson, J., Nayyeri, A.: Shortest non-crossing walks in the plane. In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 297–208. SIAM (2011)
13. Fakcharoenphol, J., Rao, S.: Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.* **72**, 868–889 (2006)
14. Frederickson, G.N.: Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.* **16**, 1004–1022 (1987)
15. Gawrychowski, P., Mozes, S., Weimann, O., Wulff-Nilsen, C.: Better tradeoffs for exact distance oracles in planar graphs. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 515–529. SIAM (2018)
16. Goldberg, A.V.: Point-to-point shortest path algorithms with preprocessing. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) *SOFSEM 2007*. LNCS, vol. 4362, pp. 88–102. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-69507-3_6

17. Gross, J.L., Tucker, T.W.: Topological Graph Theory. Courier Corporation (2001)
18. Hassin, R.: Maximum flow in (s, t) planar networks. *Inf. Process. Lett.* **13**, 107 (1981)
19. Hassin, R., Johnson, D.B.: An $O(n \log^2 n)$ algorithm for maximum flow in undirected planar networks. *SIAM J. Comput.* **14**, 612–624 (1985)
20. Henzinger, M.R., Klein, P.N., Rao, S., Subramanian, S.: Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.* **55**, 3–23 (1997)
21. Italiano, G.F., Nussbaum, Y., Sankowski, P., Wulff-Nilsen, C.: Improved algorithms for min cut and max flow in undirected planar graphs. In: Proceedings of the 43rd ACM Symposium on Theory of Computing, pp. 313–322. ACM (2011)
22. Jing, N., Huang, Y., Rundensteiner, E.A.: Hierarchical optimization of optimal path finding for transportation applications. In: CIKM 1996, Proceedings of the Fifth International Conference on Information and Knowledge Management, pp. 261–268. ACM (1996)
23. Kim, D., Maxemchuk, N.F.: Simple robotic routing in ad hoc networks. In: 13th IEEE International Conference on Network Protocols (ICNP 2005), pp. 159–168. IEEE Computer Society (2005)
24. Klein, P.N.: Multiple-source shortest paths in planar graphs. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 146–155. SIAM (2005)
25. Kowalik, L., Kurowski, M.: Short path queries in planar graphs in constant time. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing, pp. 143–148. ACM (2003)
26. Kusakari, Y., Masubuchi, D., Nishizeki, T.: Finding a noncrossing steiner forest in plane graphs under a 2-face condition. *J. Comb. Optim.* **5**, 249–266 (2001)
27. Masucci, A.P., Stanilov, K., Batty, M.: Exploring the evolution of London’s street network in the information space: a dual approach. *Phys. Rev. E* **89**, 012805 (2014)
28. Mozes, S., Sommer, C.: Exact distance oracles for planar graphs. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 209–222. SIAM (2012)
29. Nussbaum, Y.: Improved distance queries in planar graphs. In: Dehne, F., Iacono, J., Sack, J.-R. (eds.) WADS 2011. LNCS, vol. 6844, pp. 642–653. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22300-6_54
30. Raney, B., Nagel, K.: Iterative route planning for large-scale modular transportation simulations. *Future Gener. Comput. Syst.* **20**, 1101–1118 (2004)
31. Reif, J.H.: Minimum s-t cut of a planar undirected network in $O(n \log^2(n))$ time. *SIAM J. Comput.* **12**, 71–81 (1983)
32. Steiger, A.J.: Single-face non-crossing shortest paths in planar graphs. M.S. thesis, University of Illinois at Urbana-Champaign (2017). <http://hdl.handle.net/2142/98345>
33. Takahashi, J., Suzuki, H., Nishizeki, T.: Shortest Noncrossing Paths in Plane Graphs. *Algorithmica* **16**, 339–357 (1996)

34. Takahashi, J., Suzuki, H., Nishizeki, T.: Shortest non-crossing rectilinear paths in plane regions. *Int. J. Comput. Geom. Appl.* **7**, 419–436 (1997)
35. Wagner, D., Weihe, K.: A linear-time algorithm for edge-disjoint paths in planar graphs. *Combinatorica* **15**, 135–150 (1995)
36. Ziliaskopoulos, A., Kotzinos, D., Mahmassani, H.S.: Design and implementation of parallel time-dependent least time path algorithms for intelligent transportation systems applications. *Transp. Res. Part C: Emerg. Technol.* **5**, 95–107 (1997)