

Efficient Enumeration of Drawings and Combinatorial Structures for Maximal Planar Graphs

Giordano Da Lozzo¹, Giuseppe Di Battista¹, Fabrizio Frati¹,
Fabrizio Grosso², Maurizio Patrignani

¹Department of Civil Engineering, Computer Science and Aeronautical Technologies, Roma Tre University, Italy

²CeDiPa, University of Perugia, Italy

Submitted: June 2024 Accepted: March 2025 Published: May 2025

Article type: Regular paper

Communicated by: R. Uehara,
K. Yamanaka, and H.-C. Yen

Abstract. We propose efficient algorithms for enumerating the notorious combinatorial structures of maximal planar graphs, called canonical orderings and Schnyder woods, and the related classical graph drawings by de Fraysseix, Pach, and Pollack [Combinatorica, 1990] and by Schnyder [SODA, 1990], called canonical drawings and Schnyder drawings, respectively. To this aim (i) we devise an algorithm for enumerating special e -bipolar orientations of maximal planar graphs, called *canonical orientations*; (ii) we establish bijections between canonical orientations and canonical drawings, and between canonical orientations and Schnyder drawings; and (iii) we exploit the known correspondence between canonical orientations and canonical orderings, and the known bijection between canonical orientations and Schnyder woods. All our enumeration algorithms have $\mathcal{O}(n)$ setup time, space usage, and delay between any two consecutively listed outputs, for an n -vertex maximal planar graph.

Research partially funded by the European Union, Next Generation EU, Mission 4, Component 1, CUP C53D23003680006 PRIN project no. 2022TS4Y3N “EXPAND: scalable algorithms for EXPLoratory Analyses of heterogeneous and dynamic Networked Data”, and CUP J53D23007130006 PRIN project no. 2022ME9Z78 “NextGRAAL: Next-generation algorithms for constrained GRAPh visuALization”. The fourth author was also supported by Ce.Di.Pa. - PNC Programma unitario di interventi per le aree del terremoto del 2009-2016 - Linea di intervento 1 sub-misura B4 - “Centri di ricerca per l’innovazione” CUP J37G22000140001. A preliminary version of this research appeared in [27].

E-mail addresses: giordano.dalozzo@uniroma3.it (Giordano Da Lozzo) giuseppe.dibattista@uniroma3.it (Giuseppe Di Battista) fabrizio.frati@uniroma3.it (Fabrizio Frati) fabrizio.grosso@unipg.it (Fabrizio Grosso) maurizio.patrigani@uniroma3.it (Maurizio Patrignani)



This work is licensed under the terms of the [CC-BY](https://creativecommons.org/licenses/by/4.0/) license.

1 Introduction

In the late eighties, de Fraysseix, Pach, and Pollack [33, 34] and Schnyder [64] independently and almost simultaneously solved a question posed by Rosenstiehl and Tarjan [61] by proving that every maximal planar graph, and consequently every planar graph, admits a planar straight-line drawing in a $\mathcal{O}(n) \times \mathcal{O}(n)$ grid. Since resolution and size are measures of primary importance for the readability of a graph representation [36], the result by de Fraysseix, Pach, and Pollack [33, 34] and by Schnyder [64] has a central place in the graph visualization literature. It also finds heterogeneous applications in other research areas, for example in knot theory [21, 45, 46] and computational complexity [11, 43, 63].

The drawing algorithms presented by de Fraysseix, Pach, and Pollack and by Schnyder have become foundational for the graph drawing research area; see, e.g., [55, 69]. The combinatorial structures conceived for these algorithms have been used to solve a plethora of problems in graph drawing [2, 6, 8, 26, 31, 35, 39, 40, 42, 51, 56, 57] and beyond [3, 9, 16, 17, 18, 19, 24, 47, 48], for example the implementation of compact data structures for encoding planar graphs [3, 9, 24, 48] and efficient routing algorithms [18, 49]. In a nutshell, de Fraysseix, Pach, and Pollack's algorithm works iteratively, as it draws the vertices of a maximal planar graph one by one, while maintaining some geometric invariants on the boundary of the current drawing. The order of insertion of the vertices ensures that each newly added vertex is in the outer face of the already drawn graph and that such a graph is biconnected; this order is called *canonical ordering* (sometimes also *shelling order*). Schnyder's algorithm is based on a partition of the internal edges of a maximal planar graph into three trees rooted at the outer vertices of the graph and satisfying certain combinatorial properties; these trees form a so-called *Schnyder wood*. The three paths connecting each vertex to the roots of the trees define three regions of the plane, and the number of faces of the graph in such regions determines the vertex coordinates.

At first sight, canonical orderings and Schnyder woods appear to be distant concepts. However, Schnyder [64] already observed that there is a simple algorithm to obtain a Schnyder wood of a maximal planar graph G from a canonical ordering of G . The connection between the two combinatorial structures is deeper than this and it is best explained by the concept of *canonical orientation*. Given a canonical ordering π of G , the canonical orientation of G with respect to π is the directed graph obtained from G by orienting each edge away from the vertex that comes first in π . de Fraysseix and Ossona de Mendez [30] proved that there is a bijection between the canonical orientations and the Schnyder woods of G ; see also [54].

In this paper, we consider the problem of enumerating the above combinatorial structures and the corresponding graph drawings. The ones we present are, to the best of our knowledge, the first enumeration algorithms for drawings of graphs. An *enumeration algorithm* lists all the solutions of a problem, without duplicates, and then stops. Its efficiency is measured in terms of setup time, space usage, and maximum elapsed time (*delay*) between the outputs of two consecutive solutions; see, e.g., [7, 52, 62, 70]. We envisage notable applications of graph drawing enumeration algorithms with polynomial delay:

- (i) The possibility of providing a user with several alternative drawings optimizing different aesthetic criteria, giving her the possibility of selecting the most suitable for her needs; enumerating techniques may become an important tool for graph drawing software. This strategy has been used already in [12].
- (ii) Machine-Learning-based graph drawing tools exploit drawings of the same graph for their training; linear-time delay enumeration algorithms may provide a powerful fuel for such tools.

- (iii) Computer-aided systems for proving or disproving geometric and topological statements concerning graph drawings may benefit from enumeration algorithms for exploring the solution space of graph drawing problems.

After the publication of the conference version of this paper [27], the growing interest on this topic led to other publications [5, 4, 37]. Notably, in [37] it is described a way of encoding and therefore enumerating all the possible planar embeddings of a graph. The enumeration of graph orientations has a rich literature. Consider an undirected graph G with n vertices and m edges. In [25] algorithms are presented for generating the acyclic orientations of G with $\mathcal{O}(m)$ delay, the cyclic orientations with $\tilde{\mathcal{O}}(m)$ delay, and the acyclic orientations with a prescribed single source with $\mathcal{O}(nm)$ delay; see also earlier works on the same problem [10, 66]. In [14] the k -arc-connected orientations of G are enumerated with $\mathcal{O}(knm^2)$ delay. Of special interest for our paper is the enumeration of e -bipolar orientations of G . Let $e = (s, t)$ be an edge of G ; an e -bipolar orientation of G (often called st -orientation) is an acyclic orientation of G such that s and t are the only source and the only sink of the orientation. de Fraysseix, Ossona de Mendez, and Rosenstiehl [32] provided an algorithm for enumerating the e -bipolar orientations of G with polynomial delay. Setiawan and Nakano [65] showed how suitable data structures and topological properties of planar graph drawings can be used in order to bound the delay of the algorithm by de Fraysseix *et al.* to $\mathcal{O}(n)$, if G is a biconnected planar graph. The link between these algorithm and our paper resides in another result by de Fraysseix and Ossona de Mendez [30]: They proved that there exists a bijection between the canonical orientations and the bipolar orientations of G such that every internal vertex has at least two incoming edges. Our enumeration algorithm for canonical orientations follows the strategy devised by de Fraysseix *et al.* [32] and enhanced by Setiawan and Nakano [65] for enumerating bipolar orientations of biconnected planar graphs. However, the requirement that every internal vertex has at least two incoming edges dramatically increases the complexity of the problem and reveals new and, in our opinion, interesting topological properties of the desired orientations.

We present the following main results. Let G be an n -vertex maximal plane graph.

- First, we show an algorithm that enumerates all the canonical orientations of G . The algorithm works recursively. Namely, it applies one or two operations (edge contraction and edge removal) to G . Each application of an operation results in a smaller graph, whose canonical orientations are enumerated recursively and then modified into canonical orientations of G by orienting the contracted or removed edges.

In order for the recursive algorithm to have small delay, we need to apply an edge contraction or removal only if the corresponding branch of computation is going to produce at least one canonical orientation of G . We thus identify necessary and sufficient conditions for a subgraph of G to admit an orientation that can be extended to a canonical orientation of G . Further, we establish topological properties that determine whether applying an edge contraction or removal results in a graph satisfying the above conditions. Also, we design data structures that allow us to efficiently test for the satisfaction of these properties and to apply the corresponding operation if the test is successful.

- Second, as we show that canonical orderings are topological sortings of canonical orientations, our algorithm for enumerating canonical orientations allows us to obtain an algorithm that enumerates all canonical orderings of G . Furthermore, as canonical orientations are in bijection with Schnyder woods [30, Theorem 3.3], our algorithm for enumerating canonical orientations allows us to obtain an algorithm that enumerates all Schnyder woods of G .

- Third, we show that if we apply de Fraysseix, Pach, and Pollack’s algorithm with two distinct canonical orderings corresponding to the same canonical orientation, the algorithm outputs the same planar straight-line drawing of G . This is the key fact that we use in order to establish a bijection between the canonical orientations of G and the planar straight-line drawings of G produced by de Fraysseix, Pach, and Pollack’s algorithm. Together with our algorithm for the enumeration of canonical orientations, this allows us to enumerate such drawings.
- Fourth, we prove that the planar straight-line drawings of G obtained by Schnyder’s algorithm are in bijection with the Schnyder woods. This, together with the bijection between canonical orientations and Schnyder woods and together with our algorithm for the enumeration of canonical orientations, allows us to enumerate the planar straight-line drawings of G produced by Schnyder’s algorithm.

All our enumeration algorithms have $\mathcal{O}(n)$ setup time, $\mathcal{O}(n)$ space usage, and $\mathcal{O}(n)$ worst-case delay.

The paper is organized as follows. Section 2 contains basic definitions and properties. The subsequent sections show how to enumerate: canonical orientations (Section 3); canonical orderings and de Fraysseix, Pach, and Pollack drawings (Section 4); and Schnyder woods and Schnyder drawings (Section 5). Section 6 provides a comparison between our results and algorithms and those from the literature. Conclusions and open problems are in Section 7.

2 Preliminaries

In the following, we provide basic definitions and concepts.

Graphs with parallel edges. For technical reasons, we consider graphs and digraphs with parallel edges; edges with the same end-vertices are said to be *parallel*. We only consider (di)graphs without *self-loops*, i.e., edges with identical end-vertices. A graph without parallel edges is said to be *simple*. For a graph G , we denote the degree of a vertex v of G by $\deg_G(v)$. For a graph (resp. digraph) G , a *path* (resp. *directed path*) in G is a sequence $v_1, e_1, v_2, e_2, \dots, v_{k-1}, e_{k-1}, v_k$ such that $v_1, \dots, v_k \in V(G)$, such that $e_1, \dots, e_{k-1} \in E(G)$, such that e_i has end-vertices v_i and v_{i+1} (resp. e_i is directed from v_i to v_{i+1}), for $i = 1, \dots, k-1$, and such that no vertex appears twice in the sequence. A *cycle* (resp. *directed cycle*) is a path (resp. a directed path) with $v_1 = v_k$. Let \mathcal{D} be a digraph. An edge (u, v) of \mathcal{D} directed from u to v is *outgoing* u and *incoming* v . A *source* (resp. *sink*) of \mathcal{D} is a vertex with no incoming (resp. no outgoing) edges. We say that \mathcal{D} is *acyclic* if it contains no directed cycle. The *underlying graph* of \mathcal{D} is the undirected graph obtained from \mathcal{D} by ignoring the edge directions. An *orientation* of an undirected graph G is a digraph whose underlying graph is G .

Planar graphs. A *drawing* Γ of a graph maps each vertex to a point in the plane and each edge to a Jordan arc between its end-vertices. The drawing Γ is *planar* if no two edges cross, it is *straight-line* if each edge is mapped to a straight-line segment, and it is a *grid* drawing if all vertices have integer coordinates. Clearly, a graph might only admit a planar straight-line drawing if it is simple.

A planar drawing partitions the plane into connected regions, called *faces*. The only unbounded face is the *outer face*; the other (bounded) faces are *internal*. Two planar drawings of the same connected planar graph are *equivalent* if they determine the same circular order of the edges incident to each vertex. A *planar embedding* is an equivalence class of planar drawings. A *plane graph* is a planar graph equipped with a planar embedding and a designated outer face. When talking about

a subgraph G' of a plane graph G , we always assume that G' inherits a plane embedding from G ; sometimes we write *plane subgraph* to stress the fact that the subgraph has an associated plane embedding. A *maximal planar graph* is a planar graph without parallel edges to which no edge can be added without losing planarity or simplicity. A *maximal plane graph* is a maximal planar graph with a prescribed embedding. A vertex or edge of a plane graph is *internal* if it is not incident to the outer face, and it is *outer* otherwise. An internal edge is a *chord* if both its end-vertices are outer. Given a plane graph, when referring to a cycle bounding a face, we only list the subsequence formed by its vertices, as the edges are implied by the planar embedding.

Let G be a plane graph and let e be an edge of G with end-vertices u and v . The *contraction of e in G* is an operation that removes e from G and that “merges” u and v into a new vertex w . Suppose that the clockwise cyclic order of the edges incident to u is e, e_1^u, \dots, e_h^u and that the clockwise cyclic order of the edges incident to v is e, e_1^v, \dots, e_k^v . Then, the clockwise cyclic order of the edges incident to w is set to $e_1^u, \dots, e_h^u, e_1^v, \dots, e_k^v$. Suppose that, in G , there exist a vertex x , an edge $e' = (u, x)$, and an edge $e'' = (v, x)$. Then, the contraction of e in G turns e' and e'' into a pair of parallel edges incident to w and x . Also, suppose that there exists an edge e' that is parallel to e in G . Then, the contraction of e in G turns e' into a self-loop incident to w .

Connectivity. A graph is *connected* if it contains a path between any two vertices. A *cut-vertex* (resp. *separation pair*) in a graph is a vertex (resp. a pair of vertices) whose removal disconnects the graph. A graph is *biconnected* (*triconnected*) if it has no cut-vertex (resp. no separation pair). Note that a single edge or a set of parallel edges forms a biconnected graph. A *split pair* of a graph G is either a pair of adjacent vertices or a separation pair. The *components of G separated by a split pair $\{u, v\}$* are defined as follows. If e is an edge of G with end-vertices u and v , then it is a component of G separated by $\{u, v\}$; note that each parallel edge between u and v determines a distinct component of G separated by $\{u, v\}$. Also, let G_1, \dots, G_k be the connected components of $G \setminus \{u, v\}$. The subgraphs of G induced by $V(G_i) \cup \{u, v\}$, minus all parallel edges between u and v , are components of G separated by $\{u, v\}$, for $i = 1, \dots, k$. We will exploit the following.

Property 1. *Let H be a biconnected plane graph and let C be a cycle of H . Then the plane graph H_C consisting of the vertices and of the edges of H that lie in the interior or on the boundary of C is biconnected.*

Proof: In a biconnected plane graph, each face is bounded by a cycle [71]. Note that all the internal faces of H_C are also internal faces of H . Thus, since H is biconnected, these faces are bounded by cycles. Moreover, the outer face of H_C is bounded by the cycle C , by construction. By [15, Theorem 10.7], if the boundary of each face of a plane graph is a cycle, then the graph is biconnected. Therefore, H_C is biconnected. \square

Planar st-graphs. Given two vertices s and t of an undirected graph G , an orientation of G is an *st-orientation* if (i) it is acyclic and (ii) s and t are its unique source and unique sink, respectively. A digraph is a *planar st-graph* if and only if it is an *st-orientation* and it admits a planar embedding \mathcal{E} with s and t on the outer face; such a digraph together with \mathcal{E} is a *plane st-graph*. A face f is a \diamond -*face* if its boundary consists of two directed paths from a common source s_f to a common sink t_f . The following observations are well known.

Observation 1 ([28, 29, Lemma 1]). *A plane digraph with s and t on the outer face is a plane st-graph if and only if each of its faces is a \diamond -face.*

Observation 2 ([36, Lemma 4.2]). *Let G be a plane st-graph. Then, all the incoming (resp. all the outgoing) edges incident to any vertex v of G appear consecutively around v .*

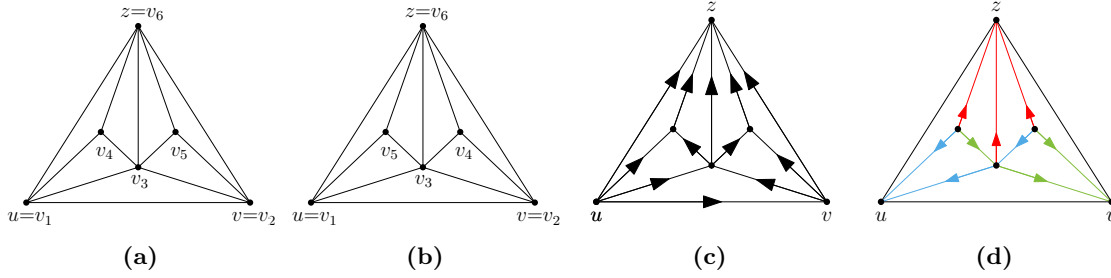


Figure 1: (a), (b) The two canonical orderings with first vertex u of a maximal plane graph G . (c) The unique canonical orientation with first vertex u of G . (d) The unique Schnyder wood of G .

Let \mathcal{D} be a plane st -graph and let $e = (u, v)$ be an edge of \mathcal{D} . The *left face* (resp. *right face*) of e is the face to the left (resp. right) of e while moving from u to v . The *left path* p_l (resp. the *right path* p_r) of \mathcal{D} consists of the edges of \mathcal{D} whose left face (resp. whose right face) is the outer face of \mathcal{D} .

In the following, let G be a maximal plane graph and let (u, v, z) be the cycle delimiting its outer face, where u , v , and z appear in this counter-clockwise order along the cycle.

Canonical orderings. A *canonical ordering* of G with first vertex u is a labeling of the vertices $v_1 = u, v_2 = v, v_3, \dots, v_{n-1}, v_n = z$ meeting the following requirements for every $k = 3, \dots, n-1$; see Figs. 1a and 1b and refer to [34].

- (CO-1) The plane subgraph $G_k \subseteq G$ induced by v_1, v_2, \dots, v_k is 2-connected; let C_k be the cycle bounding its outer face;
- (CO-2) v_{k+1} is in the outer face of G_k , and its neighbors in G_k form an (at least 2-element) subinterval of the path $C_k - (u, v)$.

A *canonical ordering* of G is a canonical ordering of G with first vertex x , where x is a vertex in $\{u, v, z\}$. Finally, if G' is a maximal planar graph, a *canonical ordering* of G' is a canonical ordering of a maximal plane graph isomorphic to G' .

Property 2. Let $\pi = (v_1 = u, v_2 = v, v_3, \dots, v_n = z)$ be a canonical ordering of a maximal plane graph G . For $i = 3, \dots, n-1$, each vertex v_i has at least two neighbors v_j with $j < i$ and one neighbor v_j with $j > i$.

Proof: Recall that, for any $i = 3, \dots, n$, we denote by G_i the plane subgraph of G induced by v_1, v_2, \dots, v_i . For $i = 4, \dots, n-1$, the fact that v_i has at least two neighbors v_j with $j < i$ directly follows by Condition (CO-2) of π . Furthermore, v_3 is adjacent to v_1 and v_2 , since G_3 is biconnected, by Condition (CO-1) of π . Suppose, for a contradiction, that, for some $i \in \{3, \dots, n-1\}$, the vertex v_i has no neighbor v_j with $j > i$. By Condition (CO-1) of π , we have that G_i is 2-connected. Also, we have that v_i is in the outer face of G_{i-1} ; this comes from Condition (CO-2) of π if $i > 3$, and from the fact that G_{i-1} is an edge if $i = 3$. Hence, v_i is incident to the outer face of G_i . Let $k \geq i$ be the largest index such that v_i is incident to the outer face of G_k . Let C_k be the cycle delimiting the outer face of G_k . Then v_{k+1} is adjacent to a vertex that comes before v_i and to a vertex that comes after v_i in the path $C_k - (u, v)$, as otherwise v_i would also be incident to the outer face of G_{k+1} , which would contradict the maximality of k . However, the fact that v_{k+1} is not adjacent to v_i implies that the neighbors of v_{k+1} in G_k do not form an interval of $C_k - (u, v)$, a contradiction to Condition (CO-2) of π . \square

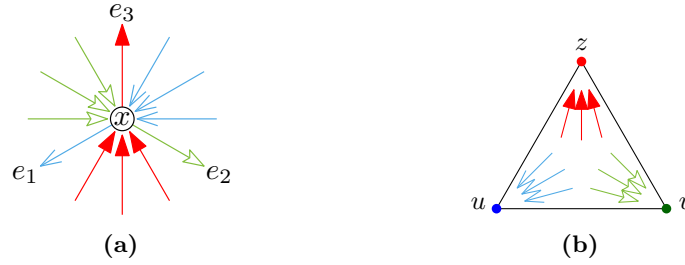


Figure 2: Representation of the two properties of a Schnyder wood. (a) Property (S-1). (b) Property (S-2).

Canonical orientations. Let $\pi = (v_1, \dots, v_n)$ be a canonical ordering of G with first vertex u . Orient every edge (v_i, v_j) of G from v_i to v_j if and only if $i < j$. The resulting orientation is the *canonical orientation of G with respect to π* . We say that an orientation \mathcal{D} of G is a *canonical orientation with first vertex u* if there exists a canonical ordering π of G with first vertex u such that \mathcal{D} is the canonical orientation of G with respect to π ; see Fig. 1c. A *canonical orientation of G* is a canonical orientation with first vertex x , where x is a vertex in $\{u, v, z\}$. Finally, if G' is a maximal planar graph, a *canonical orientation of G'* is a canonical orientation of a maximal plane graph isomorphic to G' .

Schnyder woods. A *Schnyder wood* $(\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$ of G is an assignment of directions and of the colors 1, 2 and 3 to the internal edges of G such that the following two properties hold; see Fig. 1d and refer to [64]. Let $i - 1 = 3$, if $i = 1$, and let $i + 1 = 1$, if $i = 3$.

- (S-1) For $i = 1, 2, 3$, each internal vertex x has one outgoing edge e_i of color i . The outgoing edges e_1 , e_2 , and e_3 appear in this counter-clockwise order at x . Further, for $i = 1, 2, 3$, all the incoming edges at x of color i appear in the counter-clockwise sector between the edges e_{i+1} and e_{i-1} . See Fig. 2a.
- (S-2) At the outer vertices u , v , and z , all the internal edges are incoming and of color 1, 2, and 3, respectively. See Fig. 2b.

Finally, if G' is a maximal planar graph, a *Schnyder wood of G'* is a Schnyder wood of a maximal plane graph isomorphic to G' .

3 Canonical Orientations

In [30, Lemma 3.6, Lemma 3.7, Theorem 3.3], de Fraysseix and Ossona De Mendez proved the following characterization, for which we provide here an alternative proof. Whereas the proof in [30] needs to exploit Schnyder woods, canonical orderings, and *st*-orientations in which every internal vertex has at least two incoming edges, our proof only deals with the latter two combinatorial structures.

Theorem 1 ([30]). *Let G be a maximal plane graph and let (u, v, z) be the cycle delimiting its outer face, where u , v , and z appear in this counter-clockwise order along the cycle. An orientation \mathcal{D} of G is a canonical orientation with first vertex u if and only if \mathcal{D} is a uz -orientation in which every internal vertex has at least two incoming edges.*

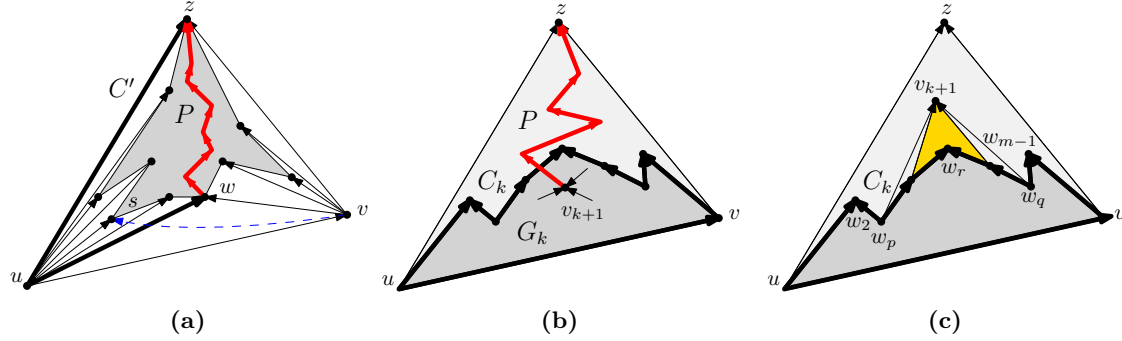


Figure 3: Illustrations for the proof of Theorem 1. (a) Illustration for the proof that $w = v_3$; the directed path P from w to z is (thick) red, the edge (v, s) is (thin dashed) blue, the edges of the cycle C' are thick. (b) Illustration for the proof that v_{k+1} lies in the outer face of G_k ; the directed path P from v_{k+1} to z is (thick) red, the edges of the cycle C_k are thick. (c) Illustration for the proof that the neighbors of v_{k+1} form a subinterval of the path $C_k - (u, v)$; the face incident to both w_r and v_{k+1} having length greater than three is shaded yellow.

Proof: (\Rightarrow) Consider any canonical orientation \mathcal{D} with first vertex u . We prove that \mathcal{D} is a uz -orientation as in the statement. Let $\pi = (v_1 = u, v_2 = v, v_3, \dots, v_{n-1}, v_n = z)$ be any canonical ordering of G such that \mathcal{D} is the canonical orientation of G with respect to π . By the construction of \mathcal{D} from π , an edge (v_i, v_j) is directed from v_i to v_j if and only if $i < j$. This implies that \mathcal{D} is an acyclic orientation, that u is a source in \mathcal{D} , and that z is a sink in \mathcal{D} . Furthermore, $v_2 = v$ has one incoming edge in \mathcal{D} , namely (v_1, v_2) , and at least one outgoing edge in \mathcal{D} , namely (v_2, v_3) . Finally, for $i = 3, 4, \dots, n-1$, by Property 2, we have that v_i has at least two incoming edges and at least one outgoing edge in \mathcal{D} .

(\Leftarrow) Let \mathcal{D} be a uz -orientation in which every internal vertex has at least two incoming edges. We prove that \mathcal{D} is a canonical orientation of G with first vertex u . Consider any topological sorting $\pi = (v_1, \dots, v_n)$ of \mathcal{D} . We show that π is a canonical ordering of G with first vertex u . Clearly, we have $v_1 = u$ and $v_n = z$, as u is the only source of \mathcal{D} and z is the only sink of \mathcal{D} . Furthermore, we have $v_2 = v$, as every vertex different from u and v has at least two incoming edges, and hence at least two vertices come before it in any topological sorting of \mathcal{D} . Let w be the vertex that is incident to an internal face of G together with u and v . We prove that every vertex of \mathcal{D} different from u and v is a successor of w , hence $v_3 = w$. Suppose, for a contradiction, that there exists a vertex $s \neq w$ that is a source of the plane digraph \mathcal{D}' obtained from \mathcal{D} by removing u and v ; refer to Fig. 3a. Consider any directed path P from w to z in \mathcal{D}' and note that s does not belong to P . Since s has at least two incoming edges in \mathcal{D} , edges from u and v to s exist. Furthermore, in \mathcal{D} , the vertex s lies either in the interior of the cycle C' bounded by the edge (u, w) , by P , and by the edge (u, z) , or in the interior of the cycle C'' bounded by the edge (v, w) , by P , and by the edge (v, z) . In the former case, the edge (v, s) crosses C' , while in the latter case, the edge (u, s) crosses C'' . In both cases, we get a contradiction to the planarity of \mathcal{D} . This contradiction proves that $v_3 = w$.

It remains to prove that, for $k = 3, \dots, n-1$, the ordering π satisfies Conditions (CO-1) and (CO-2) of a canonical ordering. In order to prove Condition (CO-1), we proceed by induction on k . In the base case, $k = 3$; that G_3 is biconnected comes from the fact that it coincides with the cycle (u, v, w) . Suppose now that G_k is biconnected, for some $k \in \{3, \dots, n-1\}$. Since v_{k+1} has at least two incoming edges, by assumption, and since the end-vertices of such edges different

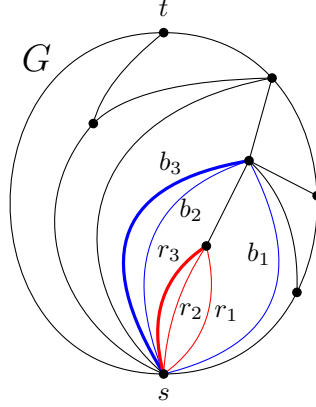


Figure 4: Illustration for the definition of a well-formed graph.

from v_{k+1} belong to G_k , since π is a topological sorting of \mathcal{D} , it follows that G_{k+1} is biconnected, which concludes the proof of Condition (CO-1). In order to prove Condition (CO-2), we first prove that, for $k = 3, \dots, n-1$, the vertex v_{k+1} is in the outer face of G_k (refer to Fig. 3b); suppose, for a contradiction, that v_{k+1} lies in the interior of C_k . Consider a directed path P in \mathcal{D} from v_{k+1} to z . Such a path exists as z is the only sink of \mathcal{D} ; moreover, P does not contain any vertex of G_k (and hence of C_k) given that π is a topological sorting of \mathcal{D} , hence every vertex v_j of P is such that $j > k$. Since v_{k+1} lies in the interior of C_k , while z lies in its exterior, by the Jordan curve's theorem we have that P crosses C_k , a contradiction which proves that v_{k+1} is in the outer face of G_k . We now prove that the neighbors of v_{k+1} in G_k form an (at least 2-element) subinterval of the path $C_k - (u, v)$; let $(w_1 = u, w_2, \dots, w_m = v)$ be such a path; refer to Fig. 3c. Let w_p and w_q be the neighbors of v_{k+1} such that p is minimum and q is maximum. Note that $p < q$, given that v_{k+1} has at least two incoming edges in \mathcal{D} ; that such edges connect v_{k+1} to vertices in C_k follows by the planarity of G . Suppose, for a contradiction, that there exists an index r with $p < r < q$ such that w_r is not a neighbor of v_{k+1} . Then the internal face of G_{k+1} incident to both v_{k+1} and w_r is also incident to w_{r-1} and w_{r+1} , hence its length is larger than three. However, since the vertices v_{k+2}, \dots, v_n and their incident edges lie in the outer face of G_{k+1} , such an internal face of G_{k+1} is also a face of G , a contradiction to the fact that G is a maximal plane graph. This concludes the proof of Condition (CO-2) and of the theorem. \square

Our proof of Theorem 1 implies the following.

Lemma 1. *Consider any canonical orientation \mathcal{D} with first vertex u of a maximal plane graph G . Then any topological sorting of \mathcal{D} is a canonical ordering of G with first vertex u .*

Proof: Let (u, v, z) be the cycle delimiting the outer face of G , where u, v , and z appear in this counter-clockwise order along the cycle. By Theorem 1, we have that \mathcal{D} is a uz -orientation in which every internal vertex has at least two incoming edges. The proof of Theorem 1 shows that every topological sorting of a uz -orientation in which every internal vertex has at least two incoming edges is a canonical ordering of G with first vertex u . \square

Given two parallel edges h_1 and h_2 with end-vertices s and x in a plane graph, we denote by $\ell(h_1, h_2)$ the open region of the plane bounded by h_1 and h_2 ; we say that $\ell(h_1, h_2)$ is a *multilens* if it contains no vertices in its interior. Observe that $\ell(h_1, h_2)$ might contain edges parallel to h_1

and h_2 in its interior, or it might coincide with an internal face of the graph. The leftmost edge of a maximal set of parallel edges is said to be *loose*, whereas the other edges of such a set are *nonloose*. In Fig. 4, any two of the (red) parallel edges r_1 , r_2 , and r_3 form a multilens, the two (blue) parallel edges b_2 and b_3 form a multilens, whereas neither b_2 nor b_3 forms a multilens with their (blue) parallel edge b_1 ; the multilenses $\ell(r_1, r_2)$, $\ell(r_2, r_3)$, and $\ell(b_2, b_3)$ are also faces; loose edges are thick (b_3 and r_3), whereas nonloose edges are thin (b_1 , b_2 , r_1 , and r_2).

The following two definitions introduce the concepts most of this section will deal with.

Definition 1. A biconnected plane graph G with two distinguished vertices s and t is called *well-formed* if it satisfies the following conditions (refer to Fig. 4):

- (WF-1) s and t are both incident to the outer face of G and s immediately precedes t in clockwise order along the cycle C_o bounding the outer face;
- (WF-2) all the internal faces of G have either two or three incident vertices;
- (WF-3) parallel edges, if any, are all incident to s ; and
- (WF-4) if there exist two parallel edges h_1 and h_2 with end-vertices s and x such that $\ell(h_1, h_2)$ is not a multilens, then there exist two parallel edges h'_1 and h'_2 between s and a vertex $y \neq x$ such that $\ell(h'_1, h'_2)$ is a multilens and such that $\ell(h'_1, h'_2) \subset \ell(h_1, h_2)$.

Vertices s and t are called *poles* of G .

Definition 2. An st -orientation \mathcal{D} of a well-formed biconnected plane graph G with poles s and t is *inner-canonical* if every internal vertex of G has at least two incoming edges in \mathcal{D} .

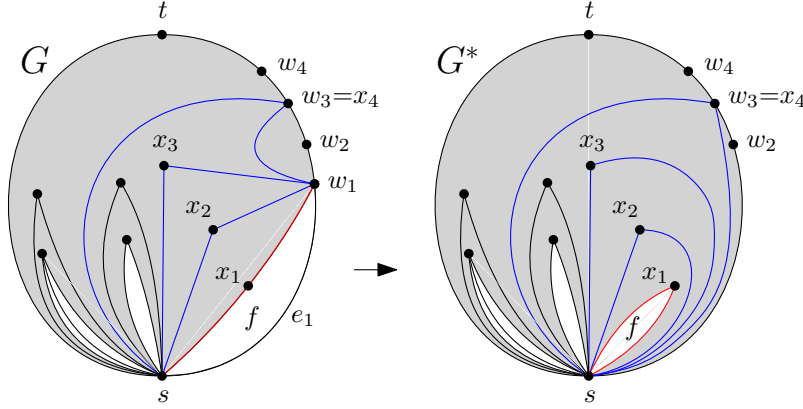
We introduce notation that will be used throughout this section. Let G be a well-formed biconnected plane graph with poles s and t . Let $(w_0 = s, w_1, \dots, w_k = t)$ be the right path p_r of G . Let e_1, e_2, \dots, e_m be the counter-clockwise order of the edges incident to s , where e_1 is the first edge of p_r and e_m is the unique edge of the left path of G , by Condition (WF-1). Let v_1, \dots, v_m be the end-vertices of e_1, \dots, e_m different from s , respectively. Moreover, denote by G^* the plane multigraph resulting from the contraction of e_1 in G . Also, if G contains parallel edges, let $j \in \{1, \dots, m-1\}$ be the smallest index such that e_j and e_{j+1} define a multilens of G ; denote by G^- the plane graph resulting from the removal of e_1, \dots, e_j from G . The next lemmata prove that, under certain conditions, G^* and G^- are well-formed multigraphs.

Lemma 2. Suppose that G does not contain parallel edges between s and w_1 . Then G^* is a well-formed biconnected plane graph with poles s and t .

Proof: Since G is biconnected, if G^* contains a cut-vertex, then this is necessarily s . It follows that $\{s, w_1\}$ is a split pair of G . Let G_1, G_2, \dots, G_k be the components separated by $\{s, w_1\}$, where $k \geq 3$ and G_1 coincides with the edge (s, w_1) of the right path of G . Since G does not contain parallel edges, we have that G_2, \dots, G_k are not single edges. Then the internal face of G that is incident to G_2 and G_3 is incident to at least four vertices, a contradiction to Condition (WF-2) of G . This proves that G^* is biconnected.

We next prove that G^* is well-formed; refer to Fig. 5.

- Condition (WF-1) follows from the fact that the left path of G^* , as well as the left path of G , is the edge (s, t) ; this trivially follows from the fact that $t \neq w_1$, since no two parallel edges between s and w_1 exist in G .

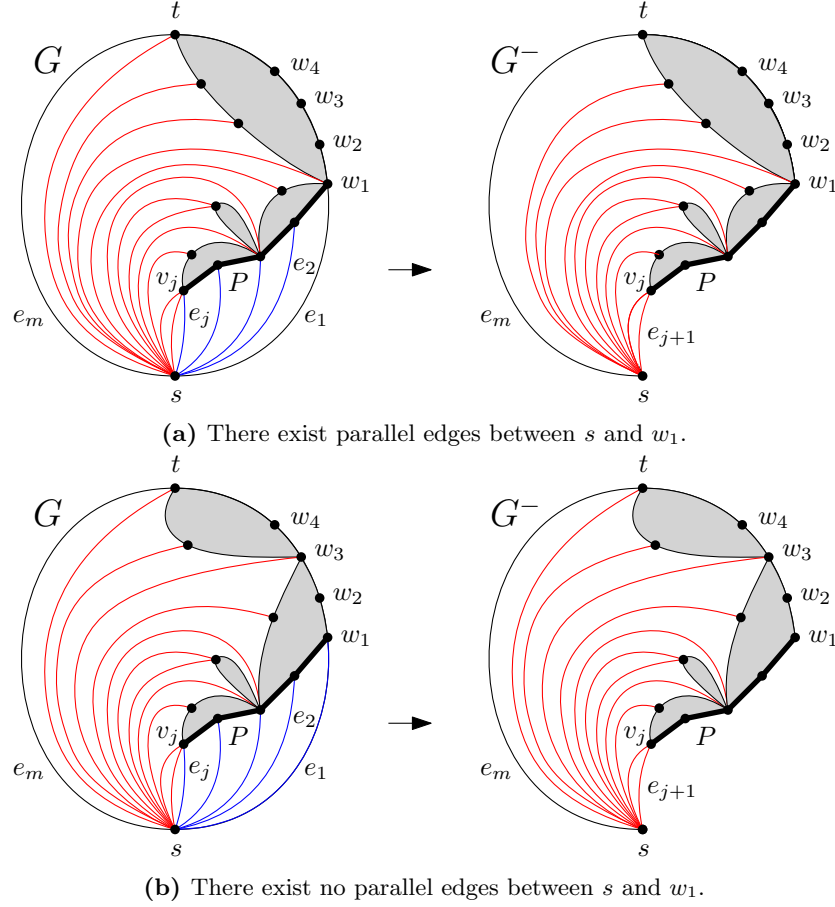
Figure 5: Illustration for the contraction of the edge e_1 .

- In order to prove Condition (WF-2), observe that G^* contains no face incident to a single vertex, as the same is true for G , by Condition (WF-2) for G , and since no two parallel edges between s and w_1 exist in G . Furthermore, that every face of G^* has at most three incident vertices descends from the fact that G satisfies Condition (WF-2) and that the contraction of an edge cannot increase the number of vertices incident to a face.
- Condition (WF-3) follows from the fact that G satisfies Condition (WF-3) and that G^* is obtained from G by the contraction of an edge that has s as an end-vertex; thus, new parallel edges, if any, are all incident to s .
- Finally, we prove Condition (WF-4). Consider any pair (e_1, e_2) of parallel edges of G^* such that $\ell(e_1, e_2)$ is not a multilens. By Condition (WF-3), both e_1 and e_2 are incident to s . If e_1 and e_2 are also parallel edges of G (that is, they do not become parallel edges because of the contraction of (s, w_1)), then there exist two parallel edges e_3 and e_4 such that $\ell(e_3, e_4)$ is a multilens and such that $\ell(e_3, e_4) \subset \ell(e_1, e_2)$ in G^* as the same is true in G , given that G satisfies Condition (WF-4). Otherwise, e_1 and e_2 are not parallel in G . Let x_1, x_2, \dots, x_k be the common neighbors of s and w_1 in G , listed in the order in which they appear in a clockwise visit of the adjacency list of w_1 , starting at the vertex x_1 that belongs to the internal face f of G incident to (s, w_1) . Let $e^\circ = (s, x_1)$ and $e^\diamond = (w_1, x_1)$ be the edges incident to f and different from (s, w_1) . Consider any pair (e_1, e_2) of parallel edges of G^* that are not parallel in G and such that $\ell(e_1, e_2)$ is not a multilens. Then we have $e_1 = (s, x_i)$ and $e_2 = (s, x_i)$, with $2 \leq i \leq k$. However, in G^* , it holds that the edges e° and e^\diamond define a multilens and that $\ell(e^\circ, e^\diamond) \subset \ell(e_1, e_2)$. Therefore Condition (WF-4) holds for G^* .

This concludes the proof that G^* is well-formed, and the proof of the lemma. \square

Lemma 3. Suppose that G contains parallel edges and let $j \in \{1, \dots, m-1\}$ be the smallest index such that e_j and e_{j+1} define a multilens of G . Suppose also that either $j = 1$, or $j > 1$ and v_2, \dots, v_j are not incident to the outer face of G . Then the graph G^- is a well-formed biconnected plane graph with poles s and t .

Proof: We start with the proof for the case in which $j = 1$. In this case, we have that e_2 is also an edge between s and w_1 . Then G^- clearly is a well-formed biconnected plane graph with

Figure 6: Illustration for the removal of the edges e_1, \dots, e_j .

poles s and t . In particular, although G^- does not contain the multilenses of G which have e_1 on their boundary, no bounded region delimited by two parallel edges contains such multilenses in its interior in G , given that e_1 is incident to the outer face of G .

We now consider the case in which $j > 1$ and v_2, \dots, v_j are not incident to the outer face of G ; refer to Fig. 6. We first prove that G^- is biconnected. In order to do that, we just need to prove that its outer face is bounded by a cycle, as the biconnectivity then follows from Property 1. By the minimality of j , for $i = 1, \dots, j$, the internal face of G delimited by e_i and e_{i+1} is triangular. Consider the plane subgraph P of G formed by the edges of such triangular faces that are not incident to s . We argue that P is a path between w_1 and v_j . Consider a clockwise Eulerian visit of the outer face of P that starts at w_1 and ends at v_j . Suppose, for a contradiction, that during this visit a vertex is encountered more than once. This implies the existence of two parallel edges e_a and e_b with $a, b \in \{1, \dots, j\}$ and with $v_a = v_b$, which contradicts the minimality of j , either directly (if e_a and e_b define a multilens) or by Condition (WF-4) (otherwise). Observe that no vertex v_i with $2 \leq i \leq j$ is incident to the outer face of G , by hypothesis. Thus, the union of the path $P \cup e_{j+1}$ and of the path $C_o - e_1$ is a cycle C_o^- , which bounds the outer face of G^- .

We next prove that G^- is well-formed.

- Condition (WF-1) follows from the fact that the left path of G^- , as well as the left path of G , is the edge (s, t) , given that $j < m$ and that e_m is the left path of G^- .
- Condition (WF-2) follows from the fact that G satisfies Condition (WF-2) and that every internal face of G^- is also an internal face of G , given that the edges e_1, e_2, \dots, e_j all lie outside C_o^- .
- Condition (WF-3) follows from the fact that G satisfies Condition (WF-3) and that the edge set of G^- is a subset of the edge set of G .
- Finally, we prove Condition (WF-4). Consider any pair (e_p, e_q) of parallel edges of G^- , where w.l.o.g. $p < q$, such that $\ell(e_p, e_q)$ is not a multilens. We prove that $\ell(e_p, e_q)$ contains a multilens in its interior in G^- . Since G^- is a subgraph of G , we have that the edges e_p and e_q also belong to G . Since G satisfies Condition (WF-4), it contains two edges e_r and e_s such that $\ell(e_r, e_s)$ is a multilens and such that $\ell(e_r, e_s) \subset \ell(e_p, e_q)$, which implies that $p \leq r \leq q$ and $p \leq s \leq q$. Since e_p belongs to G^- , it follows that $p > j$. This implies that $r > j$ and that $s > j$, hence e_r and e_s also belong to G^- and thus $\ell(e_p, e_q)$ contains a multilens in its interior in G^- .

This concludes the proof that G^- is well-formed, and the proof of the lemma. \square

Inner-canonical orientations of G^* and G^- can be used to construct inner-canonical orientations of G , as in the following two lemmata.

Lemma 4. *Let \mathcal{D}^* be an inner-canonical orientation of G^* . The orientation \mathcal{D} of G that is obtained from \mathcal{D}^* by orienting the edge (s, w_1) away from s and by keeping the orientation of all other edges unchanged is inner-canonical.*

Proof: In view of Observation 1, in order to prove that \mathcal{D} is an st -orientation, it suffices to show that all its faces are \diamond -faces. First observe that every face of G , except for the internal face f incident to (s, w_1) and for the outer face, is also a face of G^* and that its incident edges are oriented in the same way in \mathcal{D} and \mathcal{D}^* . Hence each such a face is a \diamond -face. The face f is bounded in G by the edges (s, x_1) and (s, w_1) , which are both outgoing s in \mathcal{D} , and by the edge (w_1, x_1) , which is outgoing w_1 in \mathcal{D} . Therefore f is a \diamond -face with source s and sink x_1 . The outer face of G is bounded by the edge (s, t) and the directed path s, w_1, w_2, \dots, t , which are both outgoing s in \mathcal{D} . Therefore the outer face is a \diamond -face with source s and sink t . Thus \mathcal{D} is a plane st -graph. Each internal vertex w of G is also an internal vertex of G^* , and thus it has two incoming edges in \mathcal{D} since the same property is true in \mathcal{D}^* . \square

Lemma 5. *Let \mathcal{D}^- be an inner-canonical orientation of G^- . The orientation \mathcal{D} of G that is obtained from \mathcal{D}^- by orienting the edges e_1, e_2, \dots, e_j away from s and by keeping the orientation of all other edges unchanged is inner-canonical.*

Proof: Clearly, \mathcal{D} is an st -orientation, given that \mathcal{D}^- is an st -orientation and that all the edges e_1, e_2, \dots, e_j are oriented away from the single source s of \mathcal{D}^- . Every internal vertex w of G is either an internal vertex of G^- , and thus it has two incoming edges in \mathcal{D} since the same property is true in \mathcal{D}^- , or is an end-vertex of an edge e_i , for some $i \in \{2, \dots, j\}$. In the latter case, w has at least two incoming edges in \mathcal{D} , namely e_i and at least one incoming edge it also has in \mathcal{D}^- . \square

A crucial consequence of Lemmata 2 to 5 is the following.

Lemma 6. *Every well-formed biconnected plane graph G with poles s and t has at least one inner-canonical orientation.*

Proof: The proof is by induction on the number of edges of G . In the base case, G is a single edge between s and t . Then the orientation of such an edge from s to t trivially is inner-canonical. For the inductive case, we distinguish two cases.

There exist no parallel edges between s and w_1 ; refer to Fig. 5. By Lemma 2, the plane graph G^* obtained by the contraction of $e_1 = (s, w_1)$ in G is biconnected and well-formed (with poles s and t). Thus, by induction, it admits an inner-canonical orientation \mathcal{D}^* . By Lemma 4, orienting the edge e_1 away from s and keeping the orientation of all other edges unchanged turns \mathcal{D}^* into an inner-canonical orientation \mathcal{D} of G .

There exist parallel edges between s and w_1 ; refer to Fig. 6a. In order to prove that G admits an inner-canonical orientation, it suffices to prove that the index j , defined in Lemma 3 as the smallest index such that e_j and e_{j+1} define a multilens, exists. Indeed, if such an index exists, we have that, by Lemma 3, the plane graph G^- obtained from G by removing the edges e_1, e_2, \dots, e_j is well-formed and thus, by induction, it admits an inner-canonical orientation \mathcal{D}^- . Also, by Lemma 5, orienting the edges e_1, e_2, \dots, e_j away from s turns \mathcal{D}^- into an inner-canonical orientation \mathcal{D} of G , which proves the statement.

We now show that j exists. By hypothesis, there exist two parallel edges between s and w_1 . Since e_1 connects s and w_1 , it follows that e_1 is one of such edges. Let e_h be a distinct edge also connecting s and w_1 . By Condition (WF-4), there exist two edges e_p and e_{p+1} that define a multilens and such that $1 \leq p < p+1 \leq h$. We show that choosing j as the smallest index p such that e_p and e_{p+1} define a multilens allows Lemma 3 to be applied. If $j = 1$, then Lemma 3 trivially applies. If $j > 1$, consider any edge e_i such that $1 < i \leq j$. We argue that v_i is not incident to the outer face of G , which allows Lemma 3 to be applied. Suppose the contrary, for a contradiction. We have $v_i \neq w_1$, as otherwise e_1 and e_i would be parallel edges, and thus, by Condition (WF-4), there would exist two edges e_p and e_{p+1} that define a multilens and such that $1 \leq p < p+1 \leq i$, contradicting the minimality of j . Since $v_i \neq w_1$, we have that v_i lies in the exterior of $\ell(e_1, e_h)$. From this and from the fact that e_i appears between e_h and e_1 in left-to-right order around s , we have that e_i crosses the cycle composed of e_1 and e_h , contradicting the planarity of G . \square

Sections 3.1 and 3.2 are devoted to the proof of the following main result.

Theorem 2. *Let G be a well-formed biconnected plane multigraph with m_G edges. There exists an algorithm with $\mathcal{O}(m_G)$ setup time and $\mathcal{O}(m_G)$ space usage that lists all the inner-canonical orientations of G with $\mathcal{O}(m_G)$ delay.*

Provided that Theorem 2 holds, we can prove the following.

Lemma 7. *Let G be an n -vertex maximal plane graph and let (u, v, z) be the cycle delimiting its outer face, where u , v , and z appear in this counter-clockwise order along the outer face of G . There exists an algorithm with $\mathcal{O}(n)$ setup time and $\mathcal{O}(n)$ space usage that lists all the canonical orientations of G with first vertex u with $\mathcal{O}(n)$ delay.*

Proof: Since G is a biconnected, in fact triconnected, well-formed plane graph with poles u and z , it suffices to prove that any inner-canonical orientation of G is also a canonical orientation of G with first vertex u , and vice versa. Namely, this and the fact that G has $\mathcal{O}(n)$ edges imply that the algorithm in Theorem 2 enumerates all canonical orientations of G within the stated bounds.

By Theorem 1, any canonical orientation of G with first vertex u is a uz -orientation such that every internal vertex has at least two incoming edges, hence it is an inner-canonical orientation of

G . Conversely, any inner-canonical orientation \mathcal{D} of G is also canonical. Indeed, by definition \mathcal{D} is a uz -orientation such that every internal vertex has at least two incoming edges. By Theorem 1, we have that \mathcal{D} is a canonical orientation with first vertex u . \square

Theorem 3. *Let G be an n -vertex maximal plane (resp. planar) graph. There exists an algorithm \mathcal{A}_1 (resp. \mathcal{A}_2) with $\mathcal{O}(n)$ setup time and $\mathcal{O}(n)$ space usage that lists all canonical orientations of G with $\mathcal{O}(n)$ delay.*

Proof: The algorithm \mathcal{A}_1 uses the one for the proof of Lemma 7 three times, namely once for each choice of the first vertex among the three vertices incident to the outer face of G . The algorithm \mathcal{A}_2 uses the algorithm \mathcal{A}_1 applied $4n - 8$ times; this is because there are $4n - 8$ maximal plane graphs that are isomorphic to G . Namely, the cycle (u, v, z) delimiting the outer face of a maximal plane graph isomorphic to G can be chosen among the $2n - 4$ facial cycles (of any planar drawing) of G , and the vertices can appear in counter-clockwise order u, v, z or u, z, v along the boundary of the outer face. Note that any two orientations produced by different applications of algorithm \mathcal{A}_2 differ on the source of the orientation, or on the sink of the orientation, or on the non-source and non-sink vertex incident to the outer face. \square

3.1 The Inner-Canonical Enumerator Algorithm

We are now ready to describe an algorithm that takes as input a well-formed biconnected plane graph G with poles s and t , and enumerates all its inner-canonical orientations; see Theorem 2. The algorithm, which we call INNER-CANONICAL ENUMERATOR (ICE, for short), works recursively; refer to Fig. 7. In the base case, G is the single edge $e_m = (s, t)$, and its unique inner-canonical orientation is the one in which the edge e_m is directed from s to t . Otherwise, the algorithm distinguishes four cases. In Cases 1 and 2, G contains parallel edges and e_1 is the unique edge between s and w_1 . Let $j \in \{2, \dots, m - 1\}$ be the smallest index such that e_j and e_{j+1} define a multilens of G ; note that $j > 1$ by the above assumption. In Case 1, there exists an index $i \in \{2, \dots, j\}$ such that v_i is incident to the outer face of G , while in Case 2 such an index does not exist. In Case 3, G does not contain parallel edges. Finally, in Case 4, G contains parallel edges between s and w_1 . Note that exactly one of Cases 1–4 applies to G .

- In Cases 1 and 3, **we contract** the edge (s, w_1) . Let G^* be the resulting plane graph and note that, by Lemma 2, G^* is biconnected and well-formed. Thus, the ICE algorithm can be applied recursively in order to enumerate all the inner-canonical orientations of G^* . The ICE algorithm then obtains all the inner-canonical orientations of G as follows: For every inner-canonical orientation \mathcal{D}^* of G^* , the algorithm constructs one inner-canonical orientation of G by orienting the edge (s, w_1) away from s and by keeping the orientation of all other edges unchanged, where some edges that are incident to s in G^* are instead incident to w_1 in G ; these are all outgoing s in \mathcal{D}^* and all outgoing w_1 in \mathcal{D} .
- In Case 4, **we remove** the edges e_1, e_2, \dots, e_j . Let G' be the resulting plane graph and note that, by Lemma 3, G' is biconnected and well-formed. Thus, the ICE algorithm can be applied recursively in order to enumerate all the inner-canonical orientations of G' . The ICE algorithm then obtains all the inner-canonical orientations of G as follows: For every inner-canonical orientation \mathcal{D}' of G' , the algorithm constructs one inner-canonical orientation of G by orienting the edges e_1, e_2, \dots, e_j away from s and by keeping the orientation of all other edges unchanged.

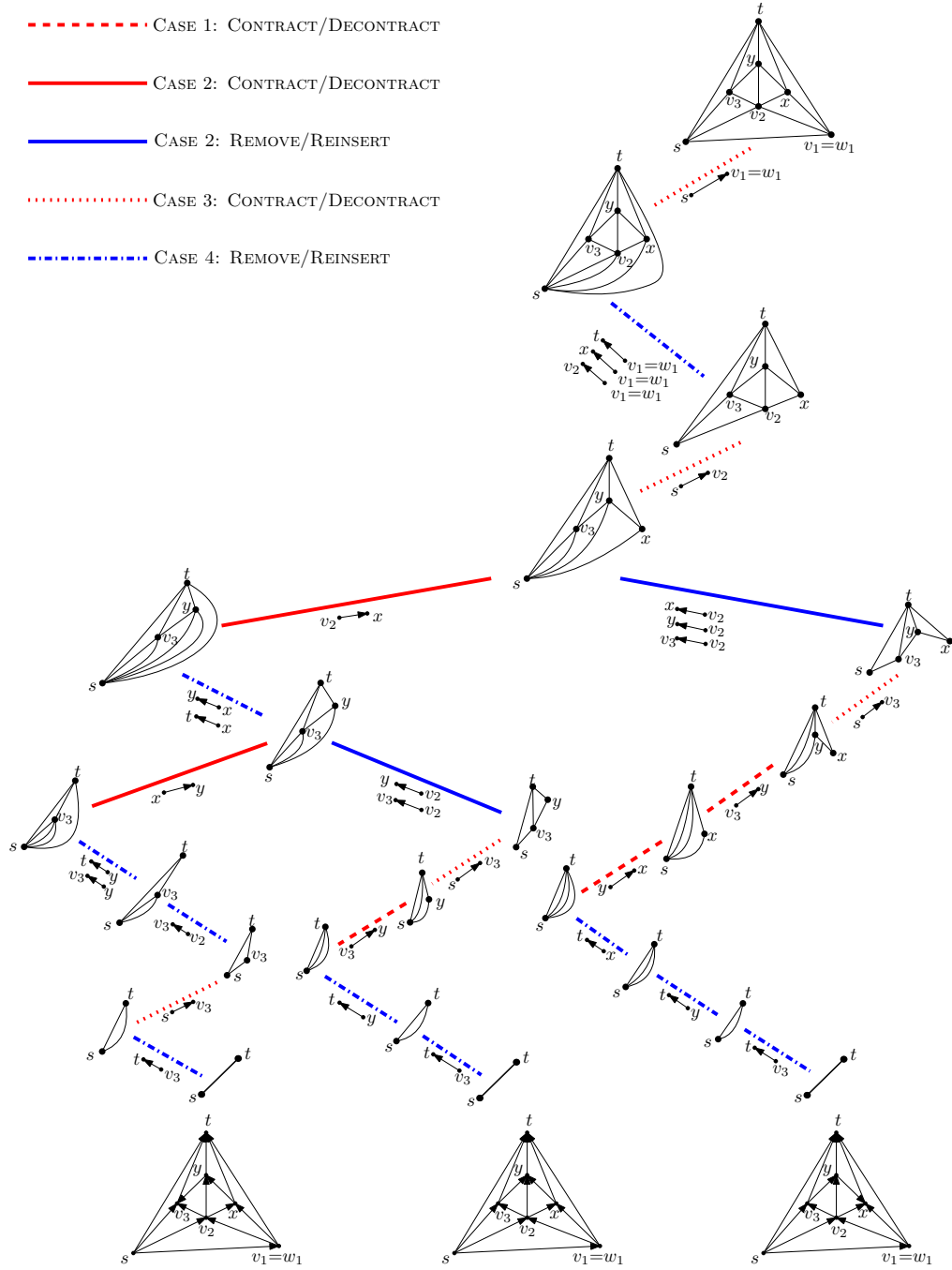


Figure 7: Illustration of the call tree of an execution of the ICE algorithm. Each of the three inner-canonical orientations of the graph in the root node is shown below the corresponding leaf node.

- In Case 2, the ICE algorithm branches and applies **both the contraction and the removal operations**. More formally, first we contract the edge (s, w_1) , obtaining a well-formed biconnected plane graph G^* , by Lemma 2. From every inner-canonical orientation \mathcal{D}^* of G^* , the algorithm constructs one inner-canonical orientation of G , same as in Cases 1 and 3. After all the inner-canonical orientations of G^* have been used to produce inner-canonical orientations of G , we remove the edges e_1, e_2, \dots, e_j from G , obtaining a well-formed biconnected plane graph G' , by Lemma 3. From every inner-canonical orientation \mathcal{D}' of G' , the algorithm constructs one inner-canonical orientation of G , same as in Case 4.

We remark that the ICE algorithm outputs an inner-canonical orientation every time the base case applies. The next three lemmata prove the correctness of the algorithm. We will later describe, in Section 3.2, how to efficiently implement it.

Lemma 8. *Every orientation of G listed by the ICE algorithm is inner-canonical.*

Proof: The proof is by induction on the size of G . The statement is trivial in the base case, hence suppose that one of Cases 1–4 applies.

In Cases 1, 2, and 3, by Lemma 2, the graph G^* constructed by the algorithm is well-formed. Hence, by induction, every orientation \mathcal{D}^* that is an output of the recursive call to ICE with input G^* is inner-canonical. Starting from \mathcal{D}^* , the ICE algorithm constructs one orientation \mathcal{D} of G by orienting the edge (s, w_1) away from s and by keeping the orientation of all other edges unchanged. By Lemma 4, we have that \mathcal{D} is inner-canonical.

In Cases 2 and 4, by Lemma 3, the graph G^- constructed by the algorithm is well-formed. Hence, by induction, every orientation \mathcal{D}^- that is an output of the recursive call to ICE with input G^- is inner-canonical. Starting from \mathcal{D}^- , the ICE algorithm constructs one orientation \mathcal{D} of G by orienting the edges e_1, e_2, \dots, e_j away from s and by keeping the orientation of all other edges unchanged. By Lemma 5, we have that \mathcal{D} is inner-canonical.

This completes the induction and hence the proof of the lemma. \square

Lemma 9. *The ICE algorithm outputs all the inner-canonical orientations of G .*

Proof: The proof is by induction on the size of G . The statement is trivial in the base case, when G is the single edge (s, t) . Otherwise, suppose that one of Cases 1–4 applies.

Suppose, for a contradiction, that there exists an inner-canonical orientation \mathcal{D} of G that is not generated by the ICE algorithm. We distinguish two cases based on the structure of G and on the orientation of the edges in \mathcal{D} . In Case A, we have that G satisfies Case 1 of the algorithm, or G satisfies Case 3 of the algorithm, or G satisfies Case 2 of the algorithm and the edge (v_1, v_2) is outgoing v_1 in \mathcal{D} ; recall that e_1, \dots, e_m is the counter-clockwise order of the edges incident to s , where e_1 is the edge in the right path of G , and that v_1, \dots, v_m are the end-vertices of e_1, \dots, e_m different from s , respectively. In Case B, we have that G satisfies Case 4 of the algorithm, or G satisfies Case 2 of the algorithm and the edge (v_1, v_2) is outgoing v_2 in \mathcal{D} . In Case A, consider the orientation \mathcal{D}^* of G^* resulting from the contraction of e_1 in \mathcal{D} . We prove below that \mathcal{D}^* is inner-canonical. Then, by induction, it is generated by the algorithm. Therefore, since by expanding the edge e_1 (as in Cases 1, 2, and 3 of the algorithm) we obtain \mathcal{D} , we get a contradiction. Analogously, in Case B, consider the orientation \mathcal{D}^- of G^- resulting from the removal of the edges e_1, e_2, \dots, e_j in \mathcal{D} , where j is the smallest index such that e_j and e_{j+1} define a multilens of G . We prove below that \mathcal{D}^- is inner-canonical. Then, by induction, it is generated by the algorithm. Therefore, since by reinserting the edges e_1, e_2, \dots, e_j (as in Cases 2 and 4 of the algorithm) we

obtain \mathcal{D} , we get a contradiction. It remains to prove that \mathcal{D}^* in Case A and \mathcal{D}^- in Case B are inner-canonical orientations.

The orientation \mathcal{D}^* is inner-canonical in Case A. First, every internal vertex of G^* has the same incident edges in \mathcal{D} and in \mathcal{D}^* (up to renaming the end-vertex of some of these edges from v_1 to s), hence every internal vertex has at least two incoming edges in \mathcal{D}^* since the same is true in \mathcal{D} . Second, we show that \mathcal{D}^* is an st -orientation of G^* . In the following, we first assume that the edge (v_1, v_2) is outgoing v_1 in \mathcal{D} and show that \mathcal{D}^* is an st -orientation of G^* under this assumption. We will then show that the edge (v_1, v_2) is indeed outgoing v_1 in \mathcal{D} .

In view of Observation 1, in order to prove that \mathcal{D}^* is an st -orientation, it suffices to show that all its faces are \diamond -faces. Let f be the internal face of G incident to e_1 . Since we are not in Case 4 of the algorithm, we have that f is a triangular face delimited by the cycle (s, v_1, v_2) . Observe that every face of G^* , except for f and for the outer face, is also a face of G and that its incident edges are oriented in the same way in \mathcal{D}^* and \mathcal{D} . Hence, each such a face is a \diamond -face of \mathcal{D}^* . The face f is bounded in G^* by two parallel edges between s and v_2 : One of them is e_2 and the other one is the edge (v_1, v_2) after the contraction that identifies v_1 and s . Both these edges are outgoing s in \mathcal{D}^* , since e_2 is outgoing s in \mathcal{D} and since (v_1, v_2) is outgoing v_1 in \mathcal{D} , by hypothesis. Therefore, f is a \diamond -face in \mathcal{D}^* with source s and sink v_2 . The outer face is bounded in G^* by the edge (s, t) and the directed path s, w_2, \dots, t , which are both outgoing s in \mathcal{D}^* . Thus, the outer face of \mathcal{D}^* is a \diamond -face with source s and sink t . This concludes the proof that \mathcal{D}^* is an st -orientation.

It remains to prove that the edge (v_1, v_2) is outgoing v_1 in \mathcal{D} . If G satisfies Case 2 of the algorithm, then the statement trivially holds true by the hypotheses of Case A. In Cases 1 and 3, let k be the smallest index such that v_k is a neighbor of s incident to the outer face; such an index exists as $v_m = t$ is incident to the outer face of G . Since in Case 3 there are no parallel edges in G and in Case 1 there are no two edges e_h and e_{h+1} defining a multilens, for any $h \in \{1, \dots, k\}$, we have that the internal faces of G delimited by e_i and e_{i+1} , for $i = 1, \dots, k-1$, are triangular. Consider the plane subgraph P of G formed by the edges of such triangular faces that are not incident to s . Such a graph is a path between v_1 and v_k (refer to the proof of Lemma 3). Suppose, for a contradiction, that the edge (v_1, v_2) is outgoing v_2 in \mathcal{D} . Consider the maximal directed subpath v_x, v_{x-1}, \dots, v_1 of P such that the edge (v_i, v_{i-1}) is outgoing v_i , for $i = x, \dots, 2$. If $x = k$, we have that \mathcal{D} contains a directed cycle formed by the path P and the subpath of p_r between v_1 and v_k , which contradicts the fact that \mathcal{D} is an st -orientation. If $x < k$, consider the edges (v_x, v_{x-1}) and (v_x, v_{x+1}) , that are both outgoing v_x in \mathcal{D} . First, (s, v_x) is incoming v_x in \mathcal{D} and it is the only edge incident to v_x that follows (v_x, v_{x+1}) and precedes (v_x, v_{x-1}) in counter-clockwise order around v_x . Second, by Observation 2, all the edges of G incident to v_x that follow (v_x, v_{x-1}) and precede (v_x, v_{x+1}) in counter-clockwise order around v_x are outgoing v_x . Hence, v_x has only one incoming edge. Since v_x is an internal vertex of G , we have a contradiction to the fact that \mathcal{D} is inner-canonical. This concludes the proof that the edge (v_1, v_2) is outgoing v_1 in \mathcal{D} in Case A.

The orientation \mathcal{D}^- is inner-canonical in Case B. First, all the internal vertices of G^- are also internal vertices of G , and thus they have two incoming edges in \mathcal{D}^- as they also do in \mathcal{D} . It remains to prove that \mathcal{D}^- is an st -orientation. With this aim, in view of Observation 1, it suffices to show that all the faces of \mathcal{D}^- are \diamond -faces. Since each internal face of \mathcal{D}^- is also an internal face of \mathcal{D} , we have that it is a \diamond -face. We now show that the outer face of \mathcal{D}^- is a \diamond -face. The left path of the outer face of G^- coincides with the edge (s, t) , hence it is a directed path from s to t . We now need to prove that the right path p_r^- of G^- is also a directed path from s to t in \mathcal{D}^- . Part of p_r^- is the subpath of p_r between v_1 and t ; this is a directed path from v_1 to t in \mathcal{D}^- , since p_r is a directed path from s to t in \mathcal{D} . If the edges e_1 and e_2 define a multilens (which implies that Case 4 applies), then p_r^- is completed with the edge e_2 , which is directed from s to v_1 in \mathcal{D}^- . Otherwise,

we have that the internal faces of G delimited by the edges e_i and e_{i+1} , for $i = 1, \dots, j-1$, are triangular. Consider the plane subgraph P of G formed by the edges of such triangular faces that are not incident to s . Such a graph is a path between v_1 and v_j (refer to the proof of Lemma 3). In order to prove that p_r^- is a directed path from s to t in \mathcal{D}^- , it suffices to prove that P is oriented from v_j to v_1 in \mathcal{D} (and thus also in \mathcal{D}^-). First, we show that P is oriented from v_j to v_1 in \mathcal{D} under the assumption that the edge (v_1, v_2) is outgoing v_2 . Then, we show that such an edge is outgoing v_2 in Case B. Consider the maximal directed subpath v_x, \dots, v_1 of P such the edge (v_i, v_{i-1}) is outgoing v_i , for $i = x, \dots, 2$. If $x = j$, we have that such a subpath coincides with P , and thus P is directed from v_j to v_1 in \mathcal{D} , as desired. Otherwise (i.e., when $1 < x < j$), we have that both the edges (v_x, v_{x-1}) and (v_x, v_{x+1}) are outgoing v_x . As in the discussion for Case A, this implies that v_x has only one incoming edge in \mathcal{D} , which is not possible since \mathcal{D} is inner-canonical. This concludes the proof that the path P is directed from v_j to v_1 in \mathcal{D} , under the assumption that the edge (v_1, v_2) is outgoing v_2 .

Next, we prove that the edge (v_1, v_2) is outgoing v_2 in Case B. In Case 2, this is true by hypothesis. In Case 4, G contains parallel edges between s and v_1 . Let e_k be the edge parallel to e_1 with the smallest index. We have that, by the construction of P , the edge (v_2, v_1) follows e_1 and precedes e_k in clockwise order around v_1 . If $v_1 = t$, then (v_2, v_1) is outgoing v_2 since \mathcal{D} is an st -orientation. Otherwise, we have that the edge $(v_1 = w_1, w_2)$ exists and is outgoing v_1 . Also, such an edge follows e_1 and precedes e_k in counter-clockwise order around v_1 . Therefore, by Observation 2, all the edges of G incident to v_1 that follow e_1 and precede e_k in clockwise order around v_1 are incoming v_1 . This concludes the proof that the edge (v_1, v_2) is outgoing v_2 in \mathcal{D} in Case B, and the proof of the lemma. \square

Lemma 10. *The ICE algorithm outputs every inner-canonical orientation of G once.*

Proof: The proof is by induction on the size of G . The statement is trivial in the base case, when G is the single edge (s, t) ; indeed, in this case no recursion is applied and hence the algorithm outputs the (unique) inner-canonical orientation of G only once.

Suppose now that G contains more than one edge. Also suppose, for a contradiction, that the algorithm produces (at least) twice the same inner-canonical orientation \mathcal{D} of G . We distinguish three cases.

First, suppose that the algorithm produces \mathcal{D} both by a “decontraction” of an inner-canonical orientation \mathcal{D}_1^* of G^* and by a decontraction of an inner-canonical orientation \mathcal{D}_2^* of G^* . We show that \mathcal{D}_1^* and \mathcal{D}_2^* are the same orientation. Indeed, \mathcal{D} is obtained (from each of \mathcal{D}_1^* and \mathcal{D}_2^*) by orienting the edge (s, w_1) away from s and by keeping the orientation of all other edges unchanged. Hence, if \mathcal{D}_1^* and \mathcal{D}_2^* were different, then also the orientations of G resulting from the decontractions of \mathcal{D}_1^* and \mathcal{D}_2^* would be different, while they are both equal to \mathcal{D} . Since \mathcal{D}_1^* and \mathcal{D}_2^* are the same orientation, by induction, the algorithm outputs such an orientation only once, hence the algorithm outputs \mathcal{D} only once, as well, a contradiction.

Second, suppose that the algorithm produces \mathcal{D} both by a “reinsertion” of directed edges in an inner-canonical orientation \mathcal{D}_1^- of G^- and by a reinsertion of directed edges in an inner-canonical orientation \mathcal{D}_2^- of G^- . We show that \mathcal{D}_1^- and \mathcal{D}_2^- are the same orientation. Indeed, \mathcal{D} is obtained (from each of \mathcal{D}_1^- and \mathcal{D}_2^-) by orienting the edges e_1, e_2, \dots, e_j away from s and by keeping the orientation of all other edges unchanged. Hence, if \mathcal{D}_1^- and \mathcal{D}_2^- were different, then also the orientations of G resulting from the reinsertion of e_1, e_2, \dots, e_j in \mathcal{D}_1^- and \mathcal{D}_2^- would be different, while they are both equal to \mathcal{D} . Since \mathcal{D}_1^- and \mathcal{D}_2^- are the same orientation, by induction, the algorithm outputs such an orientation only once, hence the algorithm outputs \mathcal{D} only once, as well, a contradiction.

Listing 1: Algorithm INNERCANONICALENUMERATOR

```

Global : Vertex  $s$  (the pole  $s$  of a well-formed graph  $G$ )
          Edge[]  $EDGES$  (the array of the edges of  $G$ )
Output : A sequence of inner-canonical orientations of  $G$ 
 $case \leftarrow \text{DetectCase}()$ 
if  $case = CONTRACT$  or  $case = CONTRACT\&REMOVE$  then
    Edge  $e_1 \leftarrow \text{Contract}()$ 
    InnerCanonicalEnumerator()
    Decontract( $e_1$ )
if  $case = REMOVE$  or  $case = CONTRACT\&REMOVE$  then
    Edge[]  $removedEdges \leftarrow \text{Remove}()$ 
    InnerCanonicalEnumerator()
    Reinsert( $removedEdges$ )
if  $case = BASE$  then
    Output()

```

Finally, suppose that the algorithm produces \mathcal{D} both by a decontraction of an inner-canonical orientation \mathcal{D}_1^* of G^* and by a reinsertion of directed edges in an inner-canonical orientation \mathcal{D}_2^- of G^- . We show that the edge (v_1, v_2) of G is oriented differently in the inner-canonical orientation \mathcal{D}^* of G resulting from the decontraction of \mathcal{D}_1^* and in the inner-canonical orientation \mathcal{D}^- of G resulting from the reinsertion of directed edges in \mathcal{D}_2^- . This contradicts the fact that \mathcal{D}^* and \mathcal{D}^- are both equal to \mathcal{D} . On the one hand, in \mathcal{D}_1^* , the vertex v_1 is identified with s , hence the edge (v_1, v_2) is outgoing v_1 . On the other hand, in \mathcal{D}_2^- , the edge (v_1, v_2) of G belongs to the right path of the outer face of G^- , with s, v_2, v_1, t in this order along such a path. Hence, the edge (v_1, v_2) is outgoing v_2 . This completes the induction and hence the proof of the lemma. \square

Lemmata 8 to 10 complete the proof of correctness of the ICE algorithm.

3.2 Efficient Implementation of the ICE Algorithm

By Lemmata 8 to 10, the ICE algorithm outputs all and only the inner-canonical orientations of G once. In the following, we show how to efficiently implement the ICE algorithm in order to achieve the stated bounds; see Theorem 2. The pseudocode of the algorithm is given in Listing 1.

In the following, we call *left-to-right order* around s the linear order of the edges incident to s obtained by visiting in clockwise order such edges starting from e_m and ending at e_1 . Analogously, we call *right-to-left order* around s the linear order of the edges incident to s obtained by visiting in counter-clockwise order such edges starting from e_1 and ending at e_m . This allows us to properly refer to an edge incident to a neighbor v_i of s as to the *rightmost* (resp. *leftmost*) edge incident to v_i of a specific type (e.g., the leftmost nonloose parallel edge incident to v_i or the rightmost parallel edge incident to v_i).

Data structures. We start by describing the data structures exploited by the algorithm. Vertices and edges of G are modelled by means of the following records (see also Listing 2).

Record of type Vertex: For each vertex x , the following information is stored:

- an integer $\deg_G(x)$;
- whether x is incident to the outer face or not;

- a reference to the rightmost edge between s and x , if any;
- if $x = s$, we also store the following information:
 - a reference to the rightmost edge e_1 incident to x ;
 - a reference to the rightmost chord incident to x , if any;
 - a reference to the rightmost parallel edge incident to x , if any; and
 - a reference to the rightmost edge belonging to a multilens, if any.

Record of type Edge: For each edge e , the following information is stored:

- a reference to the two end-vertices x and y of e ;
- whether e is oriented from x to y , or vice versa (this is initialized arbitrarily);
- whether e is an outer edge or not;
- a reference to the edge incident to x that follows e in counter-clockwise order around x and a reference to the edge incident to y that follows e in counter-clockwise order around y (this information represents the rotation system around x and y);
- if e is incident to s , we also store the following information:
 - an integer representing the position of e in left-to-right order around s (we assume that the leftmost edge (s, t) has position 1);
 - if e is parallel, a reference to the parallel edge (x, y) that follows e in right-to-left order around s , if any;
 - if e is a chord, a reference to the chord that follows e in right-to-left order around s , if any;
 - if e is parallel and nonloose, a reference to the parallel nonloose edge that follows e in right-to-left order around s , if any; and
 - if e belongs to a multilens, a reference to the nonloose edge belonging to a multilens that follows e in right-to-left order around s , if any.

The algorithm exploits the following (global) data structures:

- The input well-formed biconnected plane graph $G = (V, E)$ with poles s and t is represented using records of type **Vertex** for the vertices in V and records of type **Edge** for the edges in E . In particular, we maintain the reference to a record S of type **Vertex** corresponding to the vertex s .
- In order to efficiently output the orientation of all the edges of an inner-canonical orientation, we use an array **EDGES** whose elements are records of type **Edge** whose i -th entry contains a reference to the edge with id equal to i .

Procedures. The algorithm builds upon the procedures described below. In the remainder, we denote the type-**Vertex** record for a vertex x by $\nu(x)$ and the type-**Edge** record for an edge e by $\varepsilon(e)$. Clearly, $S = \nu(s)$.

DETECT CASE. This procedure allows us to efficiently determine which of the cases of the ICE algorithm applies to G ; refer to the pseudocode of Listing 3. We perform checks in the following order, assuming that previous checks have not concluded which case of the ICE algorithm we are in.

- We access e_1 via S and check if the reference in $\varepsilon(e_1)$ to the edge incident to s that follows e_1 in counter-clockwise order around s is **NULL**. In the positive case, we are in the *Base Case*.

Listing 2: Data Structures. Underlined pointers of a record of type **Vertex** might be different from NULL only if the vertex is s . Underlined pointers of a record of type **Edge** might be different from NULL only if one of the end-vertices of the edge is s .

```

record {
  int degree; integer representing the degree of the vertex
  bool is_outer; True if the vertex is incident to the outer face, False otherwise
  Edge first_incident_to_s; reference to the rightmost edge incident to the current vertex
    and to  $s$ ; NULL if the vertex is  $s$  or if the vertex is not adjacent to  $s$ 
  Edge  $e_1$ ; reference to the rightmost edge incident to the vertex and to  $s$ 
  Edge first_chord; reference to the rightmost chord incident to  $s$ 
  Edge first_parallel; reference to the rightmost parallel edge incident to  $s$ 
  Edge first_lens; reference to the rightmost edge of a multilens incident to  $s$ 
} Vertex;

record {
  Vertex x, y; references to the end-vertices of the edge
  bool oriented_from_x_to_y; True if the edge is oriented from  $x$  to  $y$ , False otherwise
  bool is_outer; True if the edge is an outer edge, False otherwise
  Edge next_around_x, next_around_y; Reference to the edge that follows the current edge
    in counter-clockwise order around  $x$  (around  $y$ ) in  $G$ .
  int ord; integer representing the position of the edge in the left-to-right order around  $s$ 
  Edge next_parallel_with_me; reference to the edge  $(x, y)$  that follows the current edge in
    right-to-left order around  $s$ ; not NULL only if the current edge is one of the parallel
    edges between  $x$  and  $y$ 
  Edge next_chord; reference to the chord that follows the current edge in the right-to-left
    order around  $s$ ; not NULL only if the current edge is a chord
  Edge next_nonloose_parallel; reference to the nonloose parallel edge that follows the
    current edge in right-to-left order around  $s$ ; not NULL only if the current edge is
    parallel and nonloose
  Edge next_nonloose_lens; reference to the nonloose edge belonging to a multilens that
    follows the current edge in right-to-left order around  $s$ ; not NULL only if the current
    edge belongs to a multilens
} Edge;

```

- We check if the reference in S to the rightmost parallel edge incident to s is NULL. In the positive case, we are in *Case 3*.
- We access e_1 via S and check if the reference in $\varepsilon(e_1)$ to the parallel edge with the same end-vertices as e_1 that follows e_1 in right-to-left order around s is NULL. In the negative case, we are in *Case 4*.
- Finally, we access via S the type-**Edge** record $\varepsilon(c)$ referenced by S for the rightmost chord c incident to s and the type-**Edge** record $\varepsilon(r)$ referenced by S for the rightmost edge r belonging to a multilens. If $\varepsilon(c)$ is NULL, or if the integer in $\varepsilon(c)$ representing the position of c in the left-to-right order around s is smaller than the integer in $\varepsilon(r)$ representing the position of r in the left-to-right order around s , we are in *Case 2*, otherwise we are in *Case 1*.

Clearly, we can perform all these checks in $\mathcal{O}(1)$ time.

Listing 3: Procedure DETECTCASE

Global : Vertex S ; the pole s of a well-formed graph G
Edge[] $EDGES$; the array of the edges of G

Output : A label in {BASE, CONTRACT, REMOVE, CONTRACT&REMOVE} encoding the case of the ICE algorithm that applies to G

$e_1 \leftarrow S.e_1$ The edge e_1 of G ; for simplicity of description, assume $e_1.x$ is S .
if $e_1.next_around.x$ is **NULL** **then**
 return BASE; Base Case
if $S.first_parallel$ is **NULL** **then**
 return CONTRACT; Case 3
if $e_1.next_parallel_with_me$ is not **NULL** **then**
 return REMOVE; Case 4
if $S.first_chord.ord \geq S.first_lens.ord$ **then**
 return CONTRACT; Case 1
else
 return CONTRACT&REMOVE; Case 2

OUTPUT. This procedure allows us to efficiently output an inner-canonical orientation of G . With this aim, it suffices to scan the array $EDGES$, printing for each edge its orientation. Clearly, this takes $\mathcal{O}(m_G)$ time.

CONTRACT. This procedure allows us to efficiently perform the contraction of the edge e_1 in G in order to construct the graph G^* and to update the data structures in such a way as to support the recursive calls of the ICE algorithm; refer also to the pseudocode description of Listing 4 and to Fig. 8. The procedure works as follows.

First, we access the reference to the rightmost edge e_1 incident to s via S and set the Boolean value in $\varepsilon(e_1)$ representing the orientation of such an edge to **True** if the end-vertex x of e_1 is s and to **False** otherwise. Also, we set the integer value of S representing the degree $\deg_{G^*}(s)$ of s in G^* to $\deg_G(s) + \deg_G(w_1) - 2$. Indeed, the degree of the vertex resulting from the contraction of e_1 is the sum of the degrees of the end-vertices s and w_1 of e_1 , minus two, as e_1 is incident to both s and w_1 in G and is not part of G^* . Second, the reference to the rightmost edge incident to s is set to point to the successor (w_1, w_2) of e_1 in the counter-clockwise order of the edges incident to w_1 . Third, we store for future use a reference to the rightmost parallel edge r_p incident to s in G and a reference to the rightmost chord r_c incident to s in G ; this information can be accessed via S .

Next, we perform a visit of the edges incident to w_1 in the counter-clockwise order in which they appear around w_1 in G starting at (w_1, w_2) . Throughout the visit, we keep track of the lastly visited edge l_c incident to w_1 that becomes a chord in G^* and of the lastly visited edge l_p incident to w_1 that results in a nonloose parallel edge in G^* . We execute the following actions for each encountered edge e .

Current edge's updates: We set the reference to the end-vertex of e corresponding to w_1 to point to S and the integer in $\varepsilon(e)$ representing the position of e in left-to-right order around s to be $\deg_{G^*}(s) - i + 1$, if e is the i -th edge considered in the visit.

Handling new chords: We test whether the end-vertex u of e different from w_1 (in fact, different from s , after the previous update) is incident to the outer face (this information is stored in

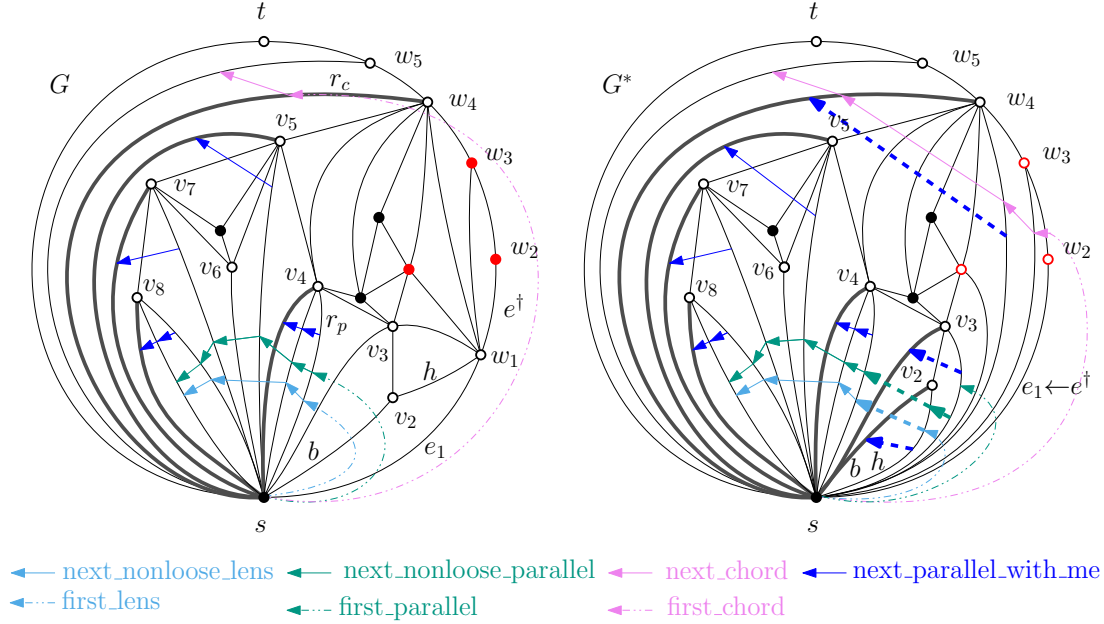


Figure 8: Illustration for the CONTRACT procedure. The input graph G is on the left, whereas the graph G^* resulting from the contraction of e_1 in G is on the right. Empty and filled circles represent vertices adjacent and not adjacent to s , respectively. The neighbors of w_1 that are not incident to S in G are red. The arrowed curves illustrate the references from type-**Edge** records to type-**Edge** records that correspond to the edges incident to s (solid or dashed) as well as the references from the type-**Vertex** record S to type-**Edge** records that correspond to the edges incident to s (dashed-dotted). To reduce clutter, the references `next_around_x` and `next_around_y` (see Listing 2) used to encode the counter-clockwise order of the edges incident to s are omitted. The references on the right that are not present on the left are dashed and thick. Loose edges are solid and thick.

$\nu(u)$). If that is the case and if $e \neq (w_1, w_2)$, we have encountered a chord of G^* . If this is the first encountered edge incident to w_1 that is a chord in G^* , then it is also the rightmost chord of G^* , hence we set the reference in S to the rightmost chord incident to s to point to $\varepsilon(e)$. Otherwise, we have already encountered an edge incident to w_1 that is a chord in G^* , and the last encountered edge of this type is stored in l_c , hence we set the reference in $\varepsilon(l_c)$ to the chord that follows l_c in the right-to-left order around s to $\varepsilon(e)$. In either case, we update l_c to e .

Handling new lenses: We test whether the edge that follows e in the counter-clockwise order of the edges incident to w_1 in G is e_1 . If this is the case, then the edge e is the edge (v_2, w_1) , labeled h in Fig. 8, and becomes the rightmost edge of a multilens of G^* composed of parallel edges between s and v_2 . Therefore, we update the reference in S to the rightmost nonloose edge of a multilens incident to s to $\varepsilon(e)$. Notice that e is the only nonloose edge that is involved in a multilens in G^* and not in G . Also, if the reference in S to the rightmost nonloose edge of a multilens incident to s used to point to an edge r_ℓ , then we update the reference in $\varepsilon(e)$ to the nonloose edge belonging to a multilens that follows e in right-to-left order around s to point to $\varepsilon(r_\ell)$.

Handling new parallel edges: We test if the end-vertex x of e different from w_1 is already adjacent to s , i.e., if x is one of the vertices v_2, \dots, v_m ; this information is stored in $\nu(x)$ as the reference to the rightmost edge e_j between x and s , for some $j \in \{2, \dots, m\}$. If this is the case, then edges between s and x exist in G , thus the contraction of e_1 turns e into an edge parallel to such edges; also, e is a nonloose edge, as it is to the right of the edges that already exist between s and x in G . We set the reference in $\varepsilon(e)$ to the edge parallel to e that follows e in right-to-left order around s to point to $\varepsilon(e_j)$. If e is the first encountered edge incident to w_1 that is a parallel edge in G^* , then it is also the rightmost parallel edge of G^* , hence we set the reference in S to the rightmost parallel edge incident to s to point to $\varepsilon(e)$. Otherwise, we have already encountered an edge incident to w_1 that is a nonloose parallel edge in G^* , and the last encountered edge of this type is stored in l_p , hence we set the reference in $\varepsilon(l_p)$ to the nonloose parallel edge that follows l_p in right-to-left order around s to $\varepsilon(e)$. In either case, we update l_p to e .

Both if edges between s and x exist in G and if they do not, the edge e is the rightmost edge incident to s and x in G^* , hence we set the reference in $\nu(x)$ to the rightmost edge incident to x and s to point to $\varepsilon(e)$.

Finally, when all the edges incident to w_1 have been visited, three more actions are performed.

First, suppose that new chords have been introduced by the contraction of e_1 . Recall that l_c is the leftmost among such chords. We update the reference in $\varepsilon(l_c)$ to the next chord in right-to-left order around s to point to $\varepsilon(r_c)$, which is the type-**Edge** record corresponding to the rightmost chord incident to s in G . This was stored before visiting the edges incident to w_1 . In this way, we link together all the chords incident to s .

Second, suppose that new nonloose parallel edges have been introduced by the contraction of e_1 . Recall that l_p is the leftmost among such edges. We update the reference in $\varepsilon(l_p)$ to the next nonloose parallel edge in right-to-left order around s to point to $\varepsilon(r_p)$, which is the type-**Edge** record corresponding to rightmost parallel edge incident to s in G . This was stored before visiting the edges incident to w_1 . In this way, we link together all non-loose parallel edges incident to s .

Third, let h be the predecessor of e_1 in counter-clockwise order around w_1 in G and let b be the successor of e_1 in G in counter-clockwise order around s ; refer to Fig. 8. We set the reference in $\varepsilon(h)$ to the edge incident to s that follows h in counter-clockwise order around s in G^* to point to $\varepsilon(b)$. In this way, we restore the rotation system around s . This concludes the description of the CONTRACT procedure. Since we visit all the edges incident to w_1 once and perform for each of them only checks and updates that take $\mathcal{O}(1)$ time, the overall procedure runs in $\mathcal{O}(\deg_G(w_1))$ time.

Listing 4: Procedure CONTRACT

Global : Vertex S ; The pole s of a well-formed graph G .
Edge[] $EDGES$; The array of the edges of G .

Output : A reference to the edge e_1 of G .

Side Effects: Sets the orientation of e_1 and contracts e_1 in G .

$e_1 \leftarrow S.e_1$; The edge e_1 of G .

Orient e_1 :

if $e_1.x$ is S **then**
 $e_1.oriented_from_x_to_y \leftarrow \text{True}$;
else
 $e_1.oriented_from_x_to_y \leftarrow \text{False}$;

In what follows, for simplicity, assume $e_1.x = s$ and $e_1.y = w_1$.
 $S.e_1 \leftarrow e_1.next_around_y$; Update S 's e_1 to be (w_1, w_2) .
 $S.degree \leftarrow S.degree + e_1.y.degree - 2$; Update S 's degree.

Counter-clockwise visit of the edges incident to w_1 :

$l_c, l_p \leftarrow \text{NULL}$; $r_p \leftarrow S.first_parallel$; $r_c \leftarrow S.first_chord$; Auxiliary references used in the visit.

$e \leftarrow S.e_1$;

for $i \leftarrow 1$; $i < e_1.y.degree$; $i \leftarrow i + 1$ **do**
 Current edge e updates (in what follows, for simplicity, assume $e.y = w_1$):
 $e.y \leftarrow S$; Identify s and w_1 .
 $e.ord \leftarrow S.degree - i + 1$
 Handling new chords:
 if $e.x.is_outer$ is *True* and $e.is_outer$ is *False* **then**
 if l_c is *NULL* **then**
 $S.first_chord \leftarrow e$; Update $S.first_chord$.
 else
 $l_c.next_chord \leftarrow e$;
 $l_c \leftarrow e$;
 Handling new lenses:
 if $e.next_around_y$ is e_1 **then**
 $r_\ell \leftarrow S.first_lens$;
 $S.first_lens \leftarrow e$;
 $e.next_nonloose_lens \leftarrow r_\ell$;
 Handling new parallel edges:
 if $e.x.first_incident_to_s$ is not *NULL* **then**
 $e.next_parallel_with_me \leftarrow e.x.first_incident_to_s$;
 if l_p is *NULL* **then**
 $S.first_parallel \leftarrow e$; Update $S.first_parallel$.
 else
 $l_p.next_nonloose_parallel \leftarrow e$;
 $l_p \leftarrow e$;
 $e.x.first_incident_to_s \leftarrow e$;
 if $i < e_1.y.degree - 1$ **then**
 $e \leftarrow e.next_around_y$; The next edge to consider in the visit.

At this point, e is the predecessor h of e_1 in counter-clockwise order around w_1 in G .

if l_c is not *NULL* **then**
 $l_c.next_chord \leftarrow r_c$;

if l_p is not *NULL* **then**
 $l_p.next_nonloose_parallel \leftarrow r_p$;

$e.next_around_y \leftarrow e_1.next_around_x$;

return e_1 ;

DECONTRACT. This procedure allows us to efficiently perform the “decontraction” of the edge e_1 in G^* , in order to obtain the graph G and the record S back. The corresponding data structures need to be updated accordingly. We omit the description of the steps of such procedure, as they can be easily deduced from the ones of the CONTRACT procedure. In particular, the edges incident to s in G^* that are incident to w_1 in G consist of the rightmost edge belonging to a multilens of G^* (whose reference is stored in S) and of all the edges that precede such an edge in the right-to-left order around s . Analogously as for the CONTRACT procedure, the DECONTRACT procedure can be implemented to run in $\mathcal{O}(\deg_G(w_1))$ time. A pseudocode description of the procedure is provided in Listing 5.

Listing 5: Procedure DECONTRACT

Global : **Vertex** S ; **The pole** s of a well-formed graph G .
Edge $[]$ $EDGES$; **The array of the edges of** G .

Input : An edge e_1

Side Effects: Sets the orientation of e_1 to NULL and decontracts e_1 in G^* in order to obtain G

$e_1.\text{oriented_from_x_to_y} \leftarrow \text{NULL}$;
 $S.\text{first_lens}$, that is h , is not part of a multilens in G .
 $h \leftarrow S.\text{first_lens}$;
 $b \leftarrow h.\text{next_parallel_with_me}$;
 leftmost_chord , leftmost_parallel ;
Visit the edges incident to s in G^* that are not incident to s in G .
 $e \leftarrow S.e_1$;
while $e \neq b$ **do**
 We assume, for simplicity, $e.x = s$, $e_1.x = s$, and $e_1.y = w_1$.
 $e.x \leftarrow e_1.y$;
 $e.y.\text{first_incident_to_s} \leftarrow e.\text{next_parallel_with_me}$; e is not incident to s in G , edges parallel to e in G^* are.
 Find the leftmost chord and the leftmost parallel edge that are not incident to s in G .
 if $e.\text{is_outer}$ is *False* and $e.y.\text{is_outer}$ is *True* **then**
 $\text{leftmost_chord} \leftarrow e$;
 if $e.\text{next_parallel_with_me}$ is not NULL **then**
 $\text{leftmost_parallel} \leftarrow e$;
 Current edge e is not incident to s in G .
 $e.\text{ord} \leftarrow \text{NULL}$;
 $e.\text{next_chord} \leftarrow \text{NULL}$;
 $e.\text{next_parallel_with_me} \leftarrow \text{NULL}$;
 $e.\text{next_nonloose_parallel} \leftarrow \text{NULL}$;
 $e.\text{next_nonloose_lens} \leftarrow \text{NULL}$;
 $S.\text{degree} \leftarrow S.\text{degree} - 1$;
 $e \leftarrow e.\text{next_around_x}$;
 $h.\text{next_around_x} \leftarrow e_1$;
 $S.\text{first_lens} \leftarrow h.\text{next_nonloose_lens}$;
 $S.\text{first_chord} \leftarrow \text{leftmost_chord}.\text{next_chord}$;
 $S.\text{first_parallel} \leftarrow \text{leftmost_parallel}.\text{next_nonloose_parallel}$;
 $S.e_1 \leftarrow e_1$;

REMOVE. This procedure allows us to efficiently perform the removal of the edges e_1, \dots, e_j (as defined in Lemma 3) in G in order to construct the graph G^- and to update the data structures in such a way as to support the recursive calls of the ICE algorithm; refer to the pseudocode description of Listing 6 and to Fig. 9. The procedure works as described next and returns an array **REMOVED**, whose entry with index i points to the type-**Edge** record $\varepsilon(e_{i+1})$ of the removed edge e_{i+1} , for $i = 0, \dots, j - 1$.

First, pointers to the type-**Edge** records corresponding to e_j and e_{j+1} are retrieved. The first one is indeed stored in the record S as the reference to the rightmost edge belonging to a multilens. The second one is instead stored in the record $\varepsilon(e_j)$ as the reference to the edge incident to s that follows e_j in counter-clockwise order around s . Second, the following updates are performed on S .

- (i) The degree of s is updated in S to be the integer stored in $\varepsilon(e_{j+1})$ representing the position of e_{j+1} in the left-to-right order around s .
- (ii) The reference in S to the rightmost edge belonging to a multilens is updated to the reference stored in $\varepsilon(e_j)$ to the nonloose edge belonging to a multilens that follows e_j in counter-clockwise order around s . Observe that, if multilenses exist in G^- , then such an edge is e_{j+1} if e_j belongs to a multilens consisting of more than two parallel edges (see Fig. 9a), whereas it is different from e_{j+1} if e_j belongs to a multilens consisting of two parallel edges (see Fig. 9b).
- (iii) The reference in S to the rightmost parallel edge incident to s is updated to the reference stored in $\varepsilon(e_j)$ to the nonloose parallel edge that follows e_j in counter-clockwise order around s . If parallel edges exist in G^- , then an analogous observation to the one given for (ii) holds also in this case; refer again to Figs. 9a and 9b.
- (iv) The reference in S to the rightmost edge incident to s is updated to point to $\varepsilon(e_{j+1})$.

Third, we perform a counter-clockwise visit of the edges incident to s to be removed, starting from e_1 and ending at e_j (both extremes are considered in the visit), and execute the following actions for each encountered edge e_i . Let x and y be the end-vertices of e_i , where $x = s$. For future use, we store the reference r_c to the rightmost chord incident to s in G , if any.

Current edge's updates: We set the Boolean value in $\varepsilon(e_i)$ representing the orientation of such an edge to **True** (i.e., e_i is oriented from $x = s$ to $y = v_i$), the Boolean value in $\nu(v_i)$ representing the fact that this vertex is incident to the outer face of G^- to **True**, and the integer value in $\nu(v_i)$ representing the degree $\deg_{G^-}(v_i)$ of v_i in G^- to $\deg_G(v_i) - 1$. Furthermore, we add e_i to the array **REMOVED** in the position with index $i - 1$.

Updating new outer edges and outer vertices: For $h = 1, \dots, j - 1$, denote by $e_{h,h+1}$ the edge (v_h, v_{h+1}) and by $e_{0,1}$ the edge $(v_1 = w_1, w_2)$ of G . We consider two cases based on whether $i \neq j$ or $i = j$.

If $i \neq j$, then e_i , e_{i+1} , and $e_{i,i+1}$ bound a triangular face of G , and (v_i, v_{i+1}) is an internal edge of G . The edges e_i and e_{i+1} do not belong to G^- , while the edge $e_{i,i+1}$ does and is an outer edge of G^- . We, therefore, set the Boolean value in $\varepsilon(e_{i,i+1})$ that represents the fact that $e_{i,i+1}$ is incident to the outer face of G^- to **True**. Also, we set the reference in $\varepsilon(e_{i,i+1})$ to the edge that follows it in counter-clockwise order around v_i to point to $\varepsilon(e_{i-1,i})$. Note that $\varepsilon(e_{i+1})$ and $\varepsilon(e_{i-1,i})$ can be accessed via $\varepsilon(e_i)$ as the record of the edge that follows e_i in counter-clockwise order around s and around v_i , respectively, and $\varepsilon(e_{i,i+1})$ can be accessed via $\varepsilon(e_{i+1})$ as the record of the edge that follows e_{i+1} in counter-clockwise order around v_{i+1} .

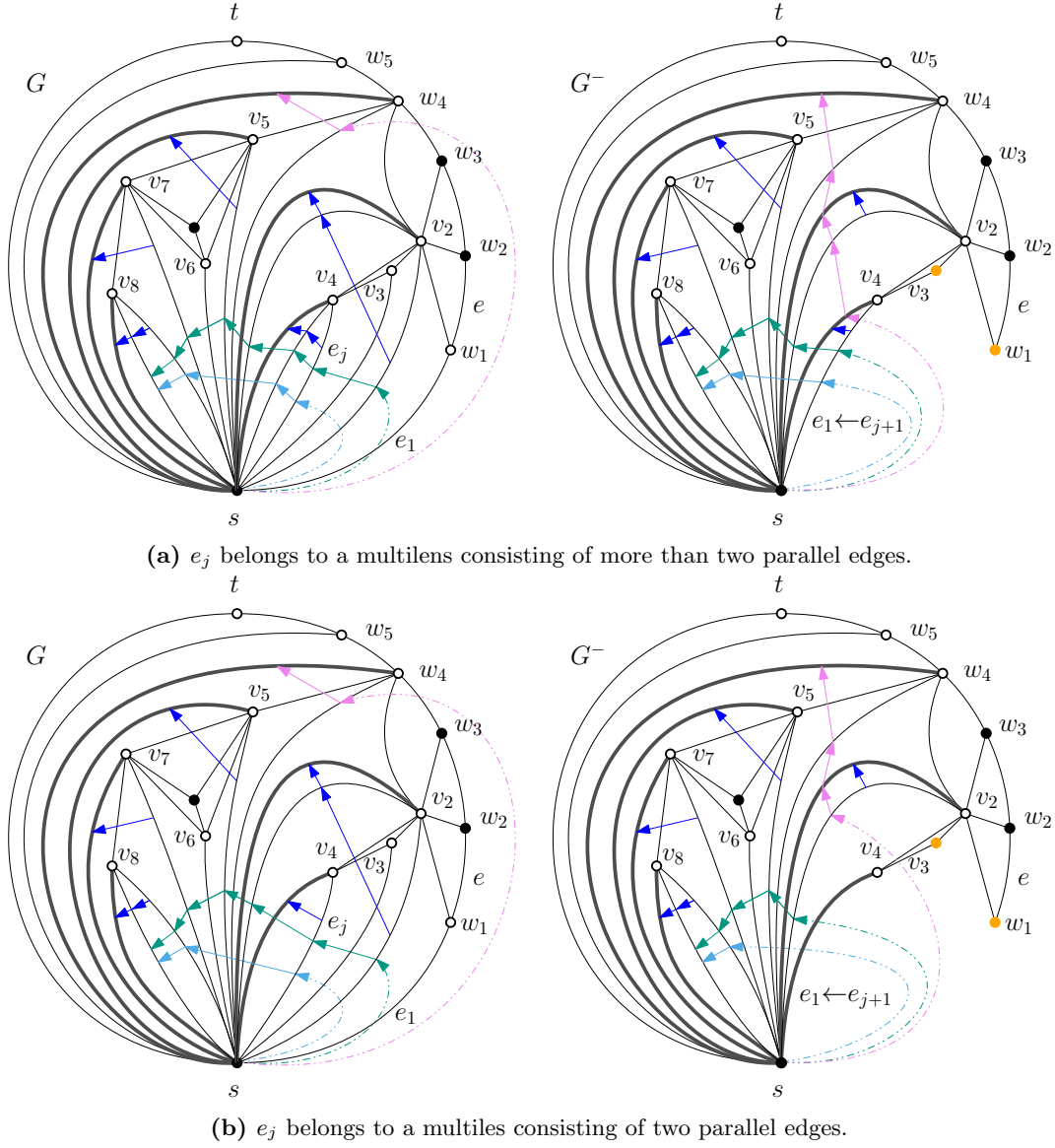


Figure 9: Illustrations for the REMOVE procedure. The input graph G is on the left, whereas the graph G^- resulting from the removal of e_1, \dots, e_j in G is on the right. The meaning of arrowed curves, of the thickness of the edges, and of the shape of the vertices is as in Fig. 8. The vertices $v_1 = w_1$ and v_3 that are neighbors of s in G but not in G^- are orange.

If $i = j$, then e_j and e_{j+1} bound an internal face of G , and in particular the edge e_{j+1} is an internal edge of G . The edge e_j does not belong to G^- , while the edge e_{j+1} does and is an outer edge of G^- . We, therefore, set the Boolean value in $\varepsilon(e_{j+1})$ that represents the fact that e_{j+1} is incident to the outer face of G^- to **True**. Also, we set the reference in $\varepsilon(e_{j+1})$ to the edge that follows it in counter-clockwise order around v_i to point to $\varepsilon(e_{j-1,j})$. Note that $\varepsilon(e_{j+1})$ can be accessed via $\varepsilon(e_i)$ as above, and $\varepsilon(e_{j-1,j})$ can be accessed via $\varepsilon(e_j)$ as the record of the edge that follows e_j in counter-clockwise order around v_j .

Updating chords: Note that the applicability of the removal operations requires that there exists no chord (s, v_h) in G with $h \leq j$. Therefore, each chord of G is also a chord of G^- . However, there may exist chords in G^- that do not belong to G . These chords, if any, are exactly the edges parallel to the removed edges e_1, \dots, e_j ; refer to Fig. 9. If neither of e_1, \dots, e_{j-1} has parallel edges and the unique edge parallel to e_j is e_{j+1} , then G^- and G have the same chords. To account for the chords in G^- that do not belong to G , we take the steps described below for each visited edge e_i .

For any $1 \leq a < b \leq j$, we detect all the chords of G^- stemming from edges parallel to e_a before the chords stemming from e_b . All the chords stemming from e_a need to be linked together. Also, if e_a and e_b each have at least one parallel edge in G and there exists no edge e_c with $a < c < b$ such that e_c has at least one parallel edge in G , then the leftmost chord of G^- stemming from e_b needs to be linked to the rightmost chord of G^- stemming from e_a . Moreover, let a' and b' be the minimum and maximum indices of edges in e_1, \dots, e_j having parallel edges. Then the leftmost chord of G^- stemming from $e_{a'}$ needs to be linked to the rightmost chord r_c of G , whereas the rightmost chord stemming from $e_{b'}$ needs to be set at the rightmost chord incident to s .

Below, we provide the details of the actions needed to implement the above updates in the data structures, when processing an edge e_i . Throughout, we maintain the record R of the rightmost encountered chord; initially, we set $R = r_c$.

- First, we access the reference to the rightmost edge r_p^i of G , if any, different from e_i that is parallel to e_i (i.e., this information is stored in $\varepsilon(e_i)$ as the reference to the edge parallel to e_i that follows e_i in right-to-left order around s in G).
- Second, we set the reference in $\nu(v_i)$ to the rightmost edge incident to v_i that is also incident to s to point to $\varepsilon(r_p^i)$. If $\varepsilon(r_p^i)$ is NULL, we proceed to consider the next edge e_{i+1} , otherwise we continue to process e_i as below.
- Third, we perform a counter-clockwise visit around s of the list of edges parallel to e_i starting from r_p^i and ending at the leftmost edge l_p^i of such a list. Let α be the currently considered edge in this visit; initially, $\alpha = r_p^i$. We set the reference in $\varepsilon(\alpha)$ to the chord that follows α in right-to-left order to point to the record of the edge β parallel to α that follows α in right-to-left order, if any. This links together all the chords stemming from e_i . When all the edges parallel to e_i have been visited, we set the reference in $\varepsilon(l_p^i)$ to the chord that follows l_p^i in right-to-left order to point to R , and then we update $R = r_p^i$. This links the chords stemming from e_i to the chords stemming from e_1, \dots, e_{i-1} and to the chords in G .

When all the edges e_1, \dots, e_j have been visited, we update the reference in S to the rightmost chord incident to S to point to R . Note that, if none of e_1, \dots, e_{j-1} has parallel edges and the unique edge parallel to e_j is e_{j+1} , then G^- and G have the same chords and $R = r_c$.

As a final step, the procedure returns the array **REMOVED**.

Since we visit all the edges e_1, \dots, e_j once and all the edges parallel to e_2, \dots, e_j once, and since we perform for each of these edges only checks and updates that take $\mathcal{O}(1)$ time, the overall procedure runs in $\mathcal{O}(j + \sum_{i=2}^j \pi(e_i))$, where $\pi(e_i)$ denotes the number of edges parallel to e_i in G . **REINSERT**. This procedure allows us to efficiently perform the “reinsertion” of the edges e_1, \dots, e_j in G^- to reconstruct the graph G , and to accordingly update the data structures. For space reasons, we omit the description of the steps of such procedure as they can be easily deduced from the ones of the **REMOVE** procedure. Analogously as for the **REMOVE** procedure, the **REINSERT** procedure can be implemented to run in $\mathcal{O}(j + \sum_{i=2}^j \pi(e_i))$. A pseudocode description of the procedure is provided in Listing 7.

We are finally ready to prove the bounds stated in Theorem 2.

Setup time: Recall that m_G denotes the number of edges of G . Initializing the type-**Vertex** and the type-**Edge** records requires $\mathcal{O}(m_G)$ time, assuming that: (i) for each vertex of G , a circularly-linked list is provided encoding the counter-clockwise order of the edges incident to v in the planar embedding of G , and that (ii) the edge (s, v_1) incident to the outer face of G is specified. Indeed, setting the type-**Vertex** and the type-**Edge** records up can be easily accomplished by suitably traversing the above lists. In particular, a first visit starting from the edge (s, v_1) allows us to determine the outer vertices and edges of G , from which the chords of G can be determined. Parallel edges can be detected easily since they are incident to s ; indeed, while traversing the list of the edges incident to s , one can mark each end-vertex different from s the first time an edge incident to it is encountered, and also keep track of a reference to that edge. Edges incident to an already marked vertex and to s are parallel edges and also make the first edge incident to those two vertices a parallel edge. The array **EDGES** can clearly be constructed in $\mathcal{O}(m_G)$ time.

Space usage: At any step, the graph considered by the ICE algorithm has at most m_G edges. Thus, the space used to represent such a structure is $\mathcal{O}(m_G)$. Also, the number of recursive calls to the ICE algorithm is $\mathcal{O}(m_G)$. Therefore, in order to show that the overall space usage of the ICE algorithm is also $\mathcal{O}(m_G)$, we only need to account for the amount of information that needs to be stored, at any moment, in the call stack, i.e., for the size of the activation records of all the calls. The top of the stack either contains the activation record of a call to the ICE algorithm on the current graph or of a call to the **OUTPUT**, **CONTRACT**, **DECONTRACT**, **REMOVE**, or **REINSERT** auxiliary procedures. The interior of the stack only contains the activation records of calls to the ICE algorithm. Whereas the activation records for each of the five auxiliary procedures are of $\mathcal{O}(1)$ size, the size of the activation record of a call to the ICE algorithm is $\mathcal{O}(1)$, if the considered call does not invoke the **REMOVE** procedure, or is $\mathcal{O}(j)$, if the considered call invokes the **REMOVE** procedure in order to remove the edges e_1, \dots, e_j . In particular, the activation record of each call to the ICE algorithm contains either a reference to the contracted edge e_1 , if the considered call invokes the **CONTRACT** procedure, or references to the removed edges e_1, \dots, e_j , if it invokes the **REMOVE** procedure. The key observation here is that a reference to an edge of G may appear only once over all the activation records that are simultaneously on the stack during the execution of the ICE algorithm, as a contracted or removed edge is not part of the graph considered in the recursive calls. Therefore, the overall space usage of the stack is $\mathcal{O}(m_G)$.

Delay: We show that the time spent by the ICE algorithm to output the first inner-canonical orientation of G is $\mathcal{O}(m_G)$, and that the time between any two inner-canonical orientations of

G that are consecutively listed by the ICE algorithm is also $\mathcal{O}(m_G)$. The recursive calls to the ICE algorithm determine a rooted binary tree \mathcal{T} , which we refer to as the *call tree*, defined as follows; see Fig. 7. The root ρ of \mathcal{T} corresponds to the first call on the input graph G , each non-root node of \mathcal{T} corresponds to the call on a graph obtained starting from G by applying a sequence of CONTRACT or REMOVE procedures. Let ν be the parent node of a node μ of \mathcal{T} , and let G_ν and G_μ be the graphs associated with ν and μ , respectively. The edge (ν, μ) either corresponds to a CONTRACT (and the symmetric DECONTRACT) procedure if $G_\mu = G_\nu^*$ or corresponds to a REMOVE (and the symmetric REINSERT) procedure if $G_\mu = G_\nu^-$.

By Lemma 6, we have that the leaves of \mathcal{T} correspond to calls to the ICE algorithm on the single edge (s, t) , which is the base case of the ICE algorithm that results in a call to the OUTPUT procedure. We consider the leaves of \mathcal{T} as ordered according to their order of creation in the construction of \mathcal{T} . Therefore, we can refer to the *first leaf* of \mathcal{T} and, given a leaf λ of \mathcal{T} , to the leaf of \mathcal{T} that *follows* λ . For each edge e of \mathcal{T} , the *cost* of e , denoted by $c(e)$, is the time spent to perform the procedure corresponding to e .

The time spent to output the first inner-canonical orientation of G coincides with the sum of the costs of all the edges of the root-to-leaf path p_α in \mathcal{T} that connects ρ and the first leaf α of \mathcal{T} , i.e., $\sum_{e \in p_\alpha} c(e)$, plus the time spent by the OUTPUT procedure. As the latter is $\mathcal{O}(m_G)$, we only need to show that the former is also $\mathcal{O}(m_G)$. As already shown in the description of the procedures, for any edge $e = (\nu, \mu)$ of \mathcal{T} , the cost $c(e)$ is at most $k_1 \cdot \deg_{G_\nu}(w_1)$, if e corresponds to a CONTRACT/DECONTRACT procedure, and is at most $k_2 \cdot (j + \sum_{i=2}^j \pi(e_i))$, if e corresponds to a REMOVE/REINSERT procedure that removes/reinserts the j rightmost edges incident to the source of G_ν , for suitable constants $k_1, k_2 > 0$. We have that $\sum_{e \in p_\alpha} c(e) \leq \max(k_1, k_2) \cdot 3m_G$. In fact, each edge of G : (1) may contribute at most twice to the degree of a vertex w_1 that has an incident edge involved in a CONTRACT/DECONTRACT procedure; (2) may appear at most once as one of the j edges removed/reinserted by a REMOVE/REINSERT procedure; and (3) might appear at most once as one of the edges parallel to some edge removed/reinserted by a REMOVE/REINSERT procedure. Indeed: (1) an edge e that is incident to w_1 in G is incident to s in G^* , hence e might only be incident to a “new” vertex w_1 involved in a second CONTRACT/DECONTRACT procedure if it is itself the edge to be contracted; after that, the edge is not part of the resulting graph G^* ; (2) an edge that appears as one of the j edges removed from G is not part of the resulting graph G^- ; and (3) if an edge e is parallel to some removed edge in e_2, \dots, e_j , then e is a chord in G^- , hence it is not a parallel edge of a removed edge later, as that would imply that a removed edge is also a chord, which does not happen in a REMOVE/REINSERT procedure.

Finally, let λ be a leaf of \mathcal{T} , let η be the leaf of \mathcal{T} that follows λ , and let ξ be the lowest common ancestor of λ and η . The time between the output of the inner-canonical orientation of G corresponding to λ and the output of the inner-canonical orientation of G corresponding to η coincides with the sum of the costs of all the edges of the path q_λ in \mathcal{T} between λ and ξ and of the costs of all the edges of the path q_η in \mathcal{T} between ξ and η , i.e., $\sum_{e \in q_\lambda} c(e) + \sum_{e \in q_\eta} c(e)$, plus the time spent by the OUTPUT procedure. As the latter is $\mathcal{O}(m_G)$, we only need to show that the former is also $\mathcal{O}(m_G)$. By the same arguments used above, we have that $\sum_{e \in q_\lambda} c(e) + \sum_{e \in q_\eta} c(e) \leq \max(k_1, k_2) \cdot 6m_G$, which is $\mathcal{O}(m_G)$.

This concludes the proof of Theorem 2.

Listing 6: Procedure REMOVE

Global : **Vertex** S ; **The pole** s of a well-formed graph G .
Edge[] $EDGES$; **The array of the edges of** G .

Output : A length- j array $REMOVED$ whose entries point to the type-**Edge** records corresponding to the edges e_1, \dots, e_j of G removed as in Lemma 3.

Side Effects: Sets the orientation of e_1, \dots, e_j and removes these edges from G .

For simplicity of description, assume $e.x$ points to S for any edge e incident to s .

$e_1 \leftarrow S.e_1$;
 $e_j \leftarrow S.first_lens$;
 $e_{j+1} \leftarrow e_j.next_around_x$;

Updating S:
 $S.degree \leftarrow e_{j+1}.ord$; **setting the degree of** S
 $S.first_lens \leftarrow e_j.next_nonloose_lens$; **setting the pointer to the rightmost edge of a multilens**
 $S.first_parallel \leftarrow e_j.next_nonloose_parallel$; **setting the pointer to the rightmost parallel edge**
 $S.e_1 \leftarrow e_{j+1}$; **setting the pointer to the rightmost edge incident to** s

Right-to-left visit of the edges incident to s to be removed:

$e \leftarrow e_1$; $r_c \leftarrow S.first_chord$; $R \leftarrow r_c$;
while e **is not** e_{j+1} **do**

Setting the orientation of $e = (s, v_i)$ **and updating the fields of** $\nu(v_i)$.
 $e.oriented_from_x_to_y \leftarrow \text{True}$; **Orient** e
 $e.y.is_outer \leftarrow \text{True}$;
 $e.y.degree \leftarrow e.y.degree - 1$;
 $REMOVED[e_1.ord - e.ord] \leftarrow e$; **Add** e **to the array of removed edges**
if e **is not** e_j **then**

Updating the incidence list of v_i **and making** (v_i, v_{i+1}) **an outer edge.**
 $e_{i,i+1} \leftarrow e.next_around_x.next_around_y$;
 $e_{i,i+1}.is_outer \leftarrow \text{True}$;
 $e_{i,i+1}.next_around_y \leftarrow e.next_around_y$; $e_{i-1,i}$ **follows** $e_{i,i+1}$ **in counterclockwise order around** v_i

else

Updating the incidence list of v_j **and making** e_{j+1} **an outer edge.**
 $e_{j+1}.is_outer \leftarrow \text{True}$;
 $e_{j+1}.next_around_y \leftarrow e_j.next_around_y$;

Updating chords:
 $r_p^i \leftarrow e.next_parallel_with_me$;
 $e.y.first_incident_to_s \leftarrow r_p^i$;

if e **is** e_j **then**

$r_p^i \leftarrow r_p^i.next_parallel_with_me$; **ignore** e_{j+1} , **which is not a chord**

if e **is not** e_1 **and** r_p^i **is not** **NULL** **then**

there are new chords (s, v_i)
 $\alpha \leftarrow r_p^i$;
 $\beta \leftarrow \alpha.next_parallel_with_me$;
while β **is not** **NULL** **do**

$\alpha.next_chord \leftarrow \beta$;
 $\alpha \leftarrow \beta$;
 $\beta \leftarrow \alpha.next_parallel_with_me$;

$\alpha.next_chord \leftarrow R$;
 $R \leftarrow r_p^i$;

if $e = e_j$ **then**

$S.first_chord \leftarrow R$; **Update the first chord**

$e \leftarrow e.next_around_x$;

return $REMOVED$;

Listing 7: Procedure REINSERT

Global : **Vertex** S ; **The pole** s of a well-formed graph G .
Edge[] $EDGES$; **The array of the edges of** G .

Input : A length- j array $REMOVED$ whose entries point to the type-**Edge** records corresponding to the edges e_1, \dots, e_j removed as in Lemma 3.

Side Effects: Reinserts the edges e_1, \dots, e_j in G^- to obtain G .

$S.e_1 \leftarrow REMOVED[0]$; **this is** e_1
 $S.first_lens \leftarrow REMOVED[REMOVED.length-1]$; **this is** e_j
 $new_first_chord \leftarrow S.first_chord$;
 $new_first_parallel \leftarrow S.first_parallel$;
for **int** $i = REMOVED.length$; $i \geq 1$; $i--$ **do**

$e_i \leftarrow REMOVED[i-1]$;
We assume, for simplicity, $e_i.x = S$, $e_i.y = v_i$
if e_i **is not** e_1 **then**
 $v_i.is_outer \leftarrow \text{False}$;
 $v_i.first_incident_to_s \leftarrow e_i$;
 $v_i.degree \leftarrow v_i.degree+1$
if e_i **is not** e_j **then**
Updating the incidence list of v_i , and making the edge (v_i, v_{i+1}) an internal edge.
We assume, for simplicity, $e_{i,i+1} = (v_i, v_{i+1})$, $e_{i,i+1}.x = v_i$, and $e_{i,i+1}.y = v_{i+1}$
 $e_{i+1} \leftarrow REMOVED[i]$;
 $e_{i,i+1} \leftarrow e_{i+1}.next_around_y$;
 $e_{i,i+1}.next_around_x \leftarrow e_i$;
 $e_{i,i+1}.is_outer \leftarrow \text{False}$;
else
Updating the incidence list of v_j and making e_{j+1} an internal edge.
 $e_{j+1} \leftarrow e_j.next_around_x$;
 $e_{j+1}.next_around_y \leftarrow e_j$;
if $e_{j+1}.next_around_x$ **is not** **NULL** **then**
 $e_{j+1}.is_outer \leftarrow \text{False}$;
Updating chords:
if e_i **is not** e_1 **and** $e_i.next_parallel_with_me$ **is not** **NULL** **then**
All the edges parallel with e_i are not chords in G .
 $e \leftarrow e_i.next_parallel_with_me$;
while $e.next_parallel_with_me$ **is not** **NULL** **do**
Both $e.next_parallel_with_me$ and $e.next_chord$ reference the next edge (s, v_i) .
 $e.next_chord \leftarrow \text{NULL}$;
 $e \leftarrow e.next_parallel_with_me$;
 $new_first_chord \leftarrow e.next_chord$;
 $e.next_chord \leftarrow \text{NULL}$;
Updating first parallel:
if $e_i.next_parallel_with_me$ **is not** **NULL** **then**
 $new_first_parallel \leftarrow e_i$;

Updating S 's first chord and first parallel
 $S.first_chord \leftarrow new_first_chord$;
 $S.first_parallel \leftarrow new_first_parallel$;
 $S.degree \leftarrow S.degree+ REMOVED.length$

4 Enumeration of Canonical Orderings and Canonical Drawings

In this section, we show how the enumeration algorithm for canonical orientations from Section 3 can be used in order to provide efficient algorithms for the enumeration of canonical orderings and canonical drawings. We start with the former.

Lemma 11. *Let G be an n -vertex maximal plane graph and let (u, v, z) be the cycle delimiting its outer face. There exists an algorithm with $\mathcal{O}(n)$ setup time and $\mathcal{O}(n)$ space usage that lists all canonical orderings of G with first vertex u with $\mathcal{O}(n)$ delay.*

Proof: The algorithm uses the algorithm in the proof of Lemma 7 for the enumeration of the canonical orientations of G with first vertex u . Indeed, for every canonical orientation \mathcal{D} of G listed by the latter algorithm, all canonical orderings π of G such that \mathcal{D} is the canonical orientation of G with respect to π can be generated as the topological sortings of \mathcal{D} , by Lemma 1. Algorithms exist for listing all such topological sortings with $\mathcal{O}(n)$ setup time, $\mathcal{O}(n)$ space usage, and even just $\mathcal{O}(1)$ delay, given that G has $\mathcal{O}(n)$ edges; see [58, 60].

All generated canonical orderings have u as first vertex since all the canonical orientations produced by the algorithms in [58, 60] have u as first vertex. Furthermore, any two canonical orderings generated from the same canonical orientation \mathcal{D} differ as any two topological sortings listed by the algorithms in [58, 60] are different from one another. Moreover, any two canonical orderings π and π' generated from different canonical orientations \mathcal{D} and \mathcal{D}' , respectively, differ as there exists an edge (w, w') in G which is oriented from w to w' in \mathcal{D} and from w' to w in \mathcal{D}' ; this implies that w precedes w' in π and follows w' in π' . \square

Theorem 4. *Let G be an n -vertex maximal plane (planar) graph. There exists an algorithm \mathcal{B}_1 (resp. \mathcal{B}_2) with $\mathcal{O}(n)$ setup time and $\mathcal{O}(n)$ space usage that lists all canonical orderings of G with $\mathcal{O}(n)$ delay.*

Proof: Algorithm \mathcal{B}_1 uses the algorithm in the proof of Lemma 11 applied three times, namely once for each choice of the first vertex among the three vertices incident to the outer face of the given maximal plane graph. Algorithm \mathcal{B}_2 uses the algorithm \mathcal{B}_1 applied $4n - 8$ times, since there are $4n - 8$ maximal plane graphs which are isomorphic to a given maximal planar graph (see the proof of Theorem 3). Note that any two canonical orderings produced by different applications of algorithm \mathcal{B}_2 differ on the first, or on the second, or on the last vertex in the ordering. \square

We now turn our attention to the enumeration of the planar straight-line drawings produced by the algorithm by de Fraysseix, Pach, and Pollack [34], known as *canonical drawings*. We begin by reviewing such an algorithm, which in the following is called *FPP algorithm*. The algorithm takes as input:

- an n -vertex maximal plane graph G , whose outer face is delimited by a cycle (u, v, z) , where u , v , and z appear in this counter-clockwise order along the outer face of G ; and
- a canonical ordering $\pi = (v_1 = u, v_2 = v, v_3, \dots, v_n = z)$ of G ; recall that G_k denotes the subgraph of G induced by the first k vertices of π .

The FPP algorithm works in $n - 2$ steps. At the first step, the FPP algorithm constructs a planar straight-line drawing Γ_3 of G_3 so that v_1 is placed at $(0, 0)$, v_2 at $(2, 0)$, and v_3 at $(1, 1)$, and defines the sets $M_3(v_1) = \{v_1, v_2, v_3\}$, $M_3(v_3) = \{v_2, v_3\}$, and $M_3(v_2) = \{v_2\}$.

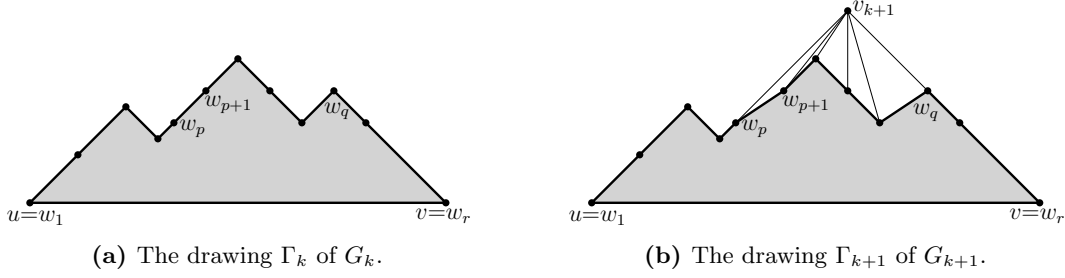


Figure 10: Illustrations for the FPP algorithm.

For any $k = 3, \dots, n-1$, at step $k-1$, the FPP algorithm constructs a planar straight-line drawing Γ_{k+1} of G_{k+1} by suitably modifying Γ_k , as follows; refer to Fig. 10. Denote by $w_1 = u, w_2, \dots, w_r = v$ the clockwise order of the vertices of G_k along its outer face. Assume that, during step $k-2$, the algorithm has defined, for $i = 1, \dots, r$, a subset $M_k(w_i)$ of the vertices of G_k such that $M_k(w_1) \supset M_k(w_2) \supset \dots \supset M_k(w_r)$. Let w_p, w_{p+1}, \dots, w_q be the neighbors of v_{k+1} in G_k , for some $1 \leq p < q \leq r$. Then Γ_{k+1} is obtained from Γ_k by increasing the x -coordinate of each vertex in $M_k(w_{p+1})$ by one unit, by increasing the x -coordinate of each vertex in $M_k(w_q)$ by one additional unit, and by placing v_{k+1} at the intersection point of the line through w_p with slope $+1$ and of the line through w_q with slope -1 . The key point for the proof of planarity of Γ_{k+1} is that the vertices w_1, \dots, w_r define in Γ_k an x -monotone path whose edges have slope either $+1$ or -1 . The “shift” of the vertices in $M_k(w_{p+1})$ makes room for drawing the edge (w_p, v_{k+1}) with slope $+1$, and the shift of the vertices in $M_k(w_q)$ makes room for drawing the edge (w_q, v_{k+1}) with slope -1 , thus maintaining the invariant on the shape of the boundary of Γ_{k+1} . Step $k-1$ is completed by defining the sets:

- $M_{k+1}(w_i) = M_k(w_i) \cup \{v_{k+1}\}$, for $i = 1, \dots, p$;
- $M_{k+1}(v_{k+1}) = M_k(w_{p+1}) \cup \{v_{k+1}\}$; and
- $M_{k+1}(w_i) = M_k(w_i)$, for $i = q, \dots, r$.

We call *canonical drawing with base edge* (u, v) the drawing Γ_n of G constructed by the FPP algorithm. We now prove the following main ingredient of our enumeration algorithm for canonical drawings.

Theorem 5. *Let G be an n -vertex maximal plane graph and let (u, v, z) be the cycle delimiting the outer face of G , where u, v , and z appear in this counter-clockwise order along the outer face of G . There exists a bijective function from the canonical orientations of G with first vertex u to the canonical drawings of G with base edge (u, v) . Also, given a canonical orientation of G with first vertex u , the corresponding canonical drawing of G with base edge (u, v) can be constructed in $\mathcal{O}(n)$ time.*

In the following we prove Theorem 5. We do this outside of a proof environment as the proof contains some statements which might be of independent interest.

We introduce some definitions. If \mathcal{D} is the canonical orientation of G with respect to π , we say that π *extends* \mathcal{D} and that π *defines* \mathcal{D} , depending on whether π is constructed from \mathcal{D} or vice versa. Also, we say that the canonical drawing Γ of G obtained by applying the FPP algorithm with a canonical ordering π of G *corresponds to* π .

The bijective function f that proves the statement of the theorem is defined as follows. Consider any canonical orientation \mathcal{D} of G with first vertex u and let π be a canonical ordering with first vertex u that extends \mathcal{D} . Then the function f maps \mathcal{D} to the canonical drawing that corresponds to π . Note that this canonical drawing has (u, v) as the base edge, since u is the first vertex of \mathcal{D} (and hence the first vertex of π) and since u, v , and z appear in this counter-clockwise order along the boundary of the outer face of G . The second part of the statement of Theorem 5 then follows from the fact that a canonical ordering that extends \mathcal{D} can be computed in $\mathcal{O}(n)$ time as any topological sorting of \mathcal{D} [50], and that the FPP algorithm can be implemented in $\mathcal{O}(n)$ time [23].

In order to prove that f is bijective, we prove that it is injective (that is, for any two distinct canonical orientations \mathcal{D}_1 and \mathcal{D}_2 of G with first vertex u , we have that $f(\mathcal{D}_1)$ and $f(\mathcal{D}_2)$ are not the same drawing) and that it is surjective (that is, for any canonical drawing Γ with base edge (u, v) , there exists a canonical orientation \mathcal{D} such that $f(\mathcal{D})$ is Γ).

We first prove that f is injective. Let \mathcal{D}_1 and \mathcal{D}_2 be any two distinct canonical orientations of G with first vertex u . For $i = 1, 2$, let π_i be any canonical ordering that extends \mathcal{D}_i . Since \mathcal{D}_1 and \mathcal{D}_2 are distinct, they differ on the orientation of some edge (a, b) different from (u, v) , say that (a, b) is directed towards b in \mathcal{D}_1 and towards a in \mathcal{D}_2 . This implies that b follows a in π_1 and precedes a in π_2 . Hence, the y -coordinate of b is larger than the one of a in $f(\mathcal{D}_1)$ and smaller than the one of a in $f(\mathcal{D}_2)$, thus $f(\mathcal{D}_1)$ and $f(\mathcal{D}_2)$ are not the same drawing.

We now prove that f is surjective. Consider any canonical drawing Γ of G with base edge (u, v) and let π be a canonical ordering of G with first vertex u such that the canonical drawing corresponding to π is Γ . Let \mathcal{D} be the canonical orientation of G with respect to π . The existence of the canonical orientation \mathcal{D} is not enough to prove that f is surjective. Indeed, given the canonical orientation \mathcal{D} , the function f considers *some* canonical ordering τ , possibly different from π , that extends \mathcal{D} , hence $f(\mathcal{D})$ is the canonical drawing corresponding to τ and it is not guaranteed that $f(\mathcal{D}) = \Gamma$. However, we have the following claim.

Claim 1. *Any two canonical orderings π and τ that extend \mathcal{D} are such that the canonical drawings of G corresponding to π and τ are the same drawing.*

Claim 1 implies that f is surjective. Indeed, the function f considers some canonical ordering τ , possibly different from π , that extends \mathcal{D} ; by Claim 1, the drawing corresponding to τ is the same drawing as the one corresponding to π , that is, Γ . It remains to prove Claim 1, which we do next. We start by extending the notions of canonical ordering, orientation, and drawing to biconnected plane graphs that are not necessarily maximal.

Let H be an m -vertex biconnected plane graph, with $m \geq 3$, whose internal faces are delimited by cycles with 3 vertices, and let u and v be two vertices incident to the outer face of H such that u immediately precedes v in the counter-clockwise order of the vertices along the boundary of the outer face of H . A *canonical ordering* of H with first vertex u is a labeling of the vertices $(v_1 = u, v_2 = v, v_3, \dots, v_{m-1}, v_m)$ such that, for $k = 3, \dots, m$, the plane subgraph H_k of H induced by v_1, v_2, \dots, v_k satisfies Conditions (CO-1) and (CO-2) in Section 2 (with H_k and H replacing G_k and G , respectively). Then a *canonical orientation* \mathcal{D}_H of H with first vertex u is obtained from a canonical ordering π of H with first vertex u by orienting each edge of H so that it is outgoing at the end-vertex that comes first in π . We say that π *extends* and *defines* \mathcal{D}_H . A *canonical drawing* Γ of H with base edge (u, v) is a drawing obtained by applying the FPP algorithm with a canonical ordering π of H . We say that Γ *corresponds* to π .

We now state the following claim, which is more general than, and hence implies, Claim 1.

Claim 2. *Any two canonical orderings π and τ that extend \mathcal{D}_H are such that the canonical drawings of H corresponding to π and τ are the same drawing.*

The proof of Claim 2 is by induction on m . The proof uses Property 3 below, which is proved inductively together with Claim 2. Let z_1, z_2, \dots, z_r be the clockwise order of the vertices along the outer face of H . Furthermore, let $M_m^\pi(z_1), M_m^\pi(z_2), \dots, M_m^\pi(z_r)$ (let $M_m^\tau(z_1), M_m^\tau(z_2), \dots, M_m^\tau(z_r)$) be the sets that are associated to the vertices z_1, z_2, \dots, z_r , respectively, by the FPP algorithm, when applied with canonical ordering π (resp. τ).

Property 3. *For $i = 1, 2, \dots, r$, the sets $M_m^\pi(z_i)$ and $M_m^\tau(z_i)$ coincide.*

Let $\pi = (u_1 = u, u_2 = v, \dots, u_m)$ and $\tau = (v_1 = u, v_2 = v, \dots, v_m)$. Also, let Γ and Φ be the canonical drawings of H that correspond to π and τ , respectively. In the base case we have $m = 3$. Then the statements of Claim 2 and Property 3 are trivial. Indeed, the first two vertices in any canonical ordering of H with first vertex u are respectively u and v , hence there is a unique canonical ordering that extends \mathcal{D}_H , there is a unique canonical drawing with base edge (u, v) , and we have $M_3^\pi(z_1) = M_3^\tau(z_1) = \{u_1, u_2, u_3\}$, $M_3^\pi(z_2) = M_3^\tau(z_2) = \{u_2, u_3\}$, and $M_3^\pi(z_3) = M_3^\tau(z_3) = \{u_2\}$.

Suppose now that $m > 3$. Assume that the statements of Claim 2 and Property 3 hold if H has $m - 1$ vertices. We prove that they also hold if H has m vertices. Let ℓ be the index such that v_ℓ is the same vertex as u_m . We observe the following simple facts.

Property 4. *We have $\ell \in \{4, \dots, m\}$.*

Proof: The first three vertices are the same in any canonical drawing with first vertex u , hence v_1, v_2 , and v_3 respectively coincide with u_1, u_2 , and u_3 and are different from u_m . \square

Property 5. *All the neighbors of v_ℓ in H precede v_ℓ in τ .*

Proof: Since u_m is the last vertex of π , all the neighbors of u_m in H precede u_m in π , hence they also precede $v_\ell = u_m$ in τ , as π and τ define the same canonical orientation of H . \square

The proof now distinguishes two cases, depending on whether $\ell = m$ or not.

Case 1: u_m and v_m are the same vertex. Let L be the $(m - 1)$ vertex plane graph obtained from H by removing the vertex u_m and its incident edges. Let w_1, w_2, \dots, w_s be the clockwise order of the vertices along the outer face of H and let w_p, w_{p+1}, \dots, w_q be the neighbors of u_m in H , for some $1 \leq p < q \leq s$. Also, let λ and ξ be the vertex orderings of L obtained from π and τ , respectively, by removing the vertex u_m . Finally, let $M_{m-1}^\lambda(w_1), M_{m-1}^\lambda(w_2), \dots, M_{m-1}^\lambda(w_s)$ (let $M_{m-1}^\xi(w_1), M_{m-1}^\xi(w_2), \dots, M_{m-1}^\xi(w_s)$) be the sets that are associated to the vertices w_1, w_2, \dots, w_s by the FPP algorithm, when applied with canonical ordering λ (resp. ξ). We next prove the following.

Lemma 12. *λ and ξ are canonical orderings of L ; furthermore, λ and ξ define the same canonical orientation of L .*

Proof: First, L is biconnected, as it coincides with the subgraph H_{m-1} of H induced by the first $m - 1$ vertices of π , which is biconnected by Condition (CO-1) of π . We show that λ is a canonical ordering of L ; the proof that ξ is also a canonical ordering of L is analogous. The first and second vertex of λ are u and v , since the same is true for π and since $m > 3$. Also, for $k = 3, \dots, m - 2$, we have that Conditions (CO-1) and (CO-2) hold for the subgraph L_k of L induced by the first k vertices in λ , given that this coincides with the subgraph H_k of H induced by the first k vertices in π and given that π is a canonical ordering of H . Finally, λ and ξ define the same canonical orientation of L , since the orientations of L defined by λ and ξ both coincide with the orientation obtained from \mathcal{D}_H by removing u_m and its incident edges. \square

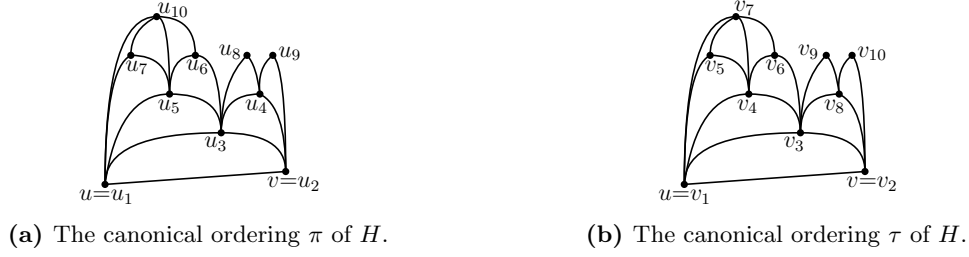


Figure 11: The case in which u_m and v_m are not the same vertex. In this example, we have $\ell = 7$. Then $\sigma_7 = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10})$, $\sigma_8 = (v_1, v_2, v_3, v_4, v_5, v_6, v_8, v_7, v_9, v_{10})$, $\sigma_9 = (v_1, v_2, v_3, v_4, v_5, v_6, v_8, v_9, v_7, v_{10})$, and $\sigma_{10} = (v_1, v_2, v_3, v_4, v_5, v_6, v_8, v_9, v_{10}, v_7)$.

By Lemma 12, we have that λ and ξ are canonical orderings of L . Let Λ and Ξ be the canonical drawings of L corresponding to λ and ξ , respectively. By induction, Λ and Ξ are the same drawing and, for $i = 1, 2, \dots, s$, we have $M_{m-1}^\lambda(w_i) = M_{m-1}^\xi(w_i)$.

The FPP algorithm constructs Γ from Λ by shifting each vertex in $M_{m-1}^\lambda(w_{p+1})$ by one unit to the right, by shifting each vertex in $M_{m-1}^\lambda(w_q)$ by one additional unit to the right, and by placing u_m at the intersection point of the line through w_p with slope $+1$, and of the line through w_q with slope -1 . Also, the FPP algorithm constructs Φ from Ξ by shifting each vertex in $M_{m-1}^\xi(w_{p+1})$ by one unit to the right, by shifting each vertex in $M_{m-1}^\xi(w_q)$ by one additional unit to the right, and by placing u_m at the intersection point of the line through w_p with slope $+1$, and of the line through w_q with slope -1 . Since Ξ coincides with Λ , since $M_{m-1}^\xi(w_{p+1}) = M_{m-1}^\lambda(w_{p+1})$, and since $M_{m-1}^\xi(w_q) = M_{m-1}^\lambda(w_q)$, we have that Φ and Γ are the same drawing, as required.

By construction, the FPP algorithm defines: (i) $M_m^\pi(w_i) = M_{m-1}^\lambda(w_i) \cup \{u_m\}$, for $i = 1, \dots, p$; (ii) $M_m^\pi(u_m) = M_{m-1}^\lambda(w_{p+1}) \cup \{u_m\}$; (iii) $M_m^\pi(w_i) = M_{m-1}^\lambda(w_i)$, for $i = q, \dots, s$; (iv) $M_m^\tau(w_i) = M_{m-1}^\xi(w_i) \cup \{u_m\}$, for $i = 1, \dots, p$; (v) $M_m^\tau(u_m) = M_{m-1}^\xi(w_{p+1}) \cup \{u_m\}$; and (vi) $M_m^\tau(w_i) = M_{m-1}^\xi(w_i)$, for $i = q, \dots, s$. Since $M_{m-1}^\xi(w_i) = M_{m-1}^\lambda(w_i)$, for $i = 1, \dots, s$, it follows that $M_m^\pi(z_i) = M_m^\tau(z_i)$, for $i = 1, \dots, r$, as required.

Case 2: u_m and v_m are not the same vertex. In this case, by Property 4, we have that u_m coincides with v_ℓ , for some $\ell \in \{4, \dots, m-1\}$. Our proof uses a sequence of canonical orderings $\sigma_{\ell+1}, \sigma_{\ell+2}, \dots, \sigma_m$ of H , where for $j = \ell+1, \dots, m$, the ordering σ_j is defined as $(v_1, \dots, v_{\ell-1}, v_{\ell+1}, \dots, v_j, v_\ell, v_{j+1}, \dots, v_m)$. That is, the order of the vertices of H in σ_j coincides with the one in τ , except that v_ℓ is shifted to the j -th position. See Fig. 11 for an example. We first prove that $\sigma_{\ell+1}, \sigma_{\ell+2}, \dots, \sigma_m$ are indeed canonical orderings of H .

Lemma 13. For $j = \ell+1, \dots, m$, we have that σ_j is a canonical ordering of H ; furthermore, the canonical orientation of H defined by σ_j is \mathcal{D}_H .

Proof: First, since u_m coincides with v_ℓ , for some $\ell \in \{4, \dots, m-1\}$, the first and second vertex of σ_j are u and v , since the same is true for τ . For $k = 3, \dots, m$, let H_k^τ and H_k^j be the plane subgraphs induced by the first k vertices of H in τ and in σ_j , respectively. We now prove that H_k^j satisfies Conditions (CO-1) and (CO-2), for $k = 3, \dots, m-1$.

- *Condition (CO-1).* For $k = 3, \dots, \ell-1, j, j+1, \dots, m-1$, we have that H_k^j is biconnected, because it coincides with H_k^τ , and because H_k^τ satisfies Condition (CO-1). We next prove that the graphs $H_\ell^j, H_{\ell+1}^j, \dots, H_{j-1}^j$ are biconnected. The vertices $v_{\ell+1}, v_{\ell+2}, \dots, v_j$ are not

neighbors of v_ℓ in H , since by Property 5 all the neighbors of v_ℓ in H precede v_ℓ in τ and hence are among $v_1, v_2, \dots, v_{\ell-1}$. Also, for $k = \ell, \ell+1, \dots, j-1$, we have that H_k^j is the graph obtained from H_{k+1}^τ by removing v_ℓ and its incident edges. Hence, for $k = \ell-1, \ell, \dots, j-2$, the $(k+1)$ -th vertex of σ_j has at least two neighbors in H_k^j , given that it coincides with the $(k+2)$ -th vertex of τ , which has at least two neighbors in H_{k+1}^τ since H_{k+1}^τ satisfies Condition (CO-2), and given that it is not a neighbor of v_ℓ . This, together with the fact that $H_{\ell-1}^j$ is biconnected, implies the biconnectivity of $H_\ell^j, H_{\ell+1}^j, \dots, H_{j-1}^j$.

- *Condition (CO-2).* For $k = 3, \dots, \ell-2, j, j+1, \dots, m-1$, the $(k+1)$ -th vertex of σ_j is in the outer face of H_k^j because it is the same vertex as the $(k+1)$ -th vertex of τ , because H_k^j coincides with H_k^τ , and because the $(k+1)$ -th vertex of τ is in the outer face of H_k^τ , given that H_k^τ satisfies Condition (CO-2). For $k = \ell-1, \ell, \dots, j-2$, the $(k+1)$ -th vertex of σ_j is in the outer face of H_k^j because it is the same vertex as the $(k+2)$ -th vertex of τ , because H_k^j is a subgraph of H_{k+1}^τ , and because the $(k+2)$ -th vertex of τ is in the outer face of H_{k+1}^τ , given that H_{k+1}^τ satisfies Condition (CO-2). Finally, the j -th vertex of σ_j , that is $v_\ell = u_m$, is in the outer face of H_{j-1}^j because it is in the outer face of the plane subgraph H_{m-1} of H induced by the first $m-1$ vertices of π , given that H_{m-1} satisfies Condition (CO-2).

This proves that σ_j is a canonical ordering. The fact that the canonical orientation \mathcal{D}_j defined by σ_j is \mathcal{D}_H follows from the following facts: (i) the canonical orientation defined by τ is \mathcal{D}_H ; (ii) the orderings τ and σ_j coincide on the vertices of H different from v_ℓ – this implies that the orientation of every edge that is not incident to v_ℓ is the same in \mathcal{D}_j and in \mathcal{D}_H ; and (iii) the vertex v_ℓ follows its neighbors in H both in τ and in σ_j – this implies that the orientation of every edge incident to v_ℓ is the same in \mathcal{D}_j and in \mathcal{D}_H . \square

In order to simplify the description, let σ_ℓ coincide with τ . For $j = \ell, \dots, m$, let Σ^j be the canonical drawing of H that corresponds to σ_j . Also, let $M_m^j(z_1), M_m^j(z_2), \dots, M_m^j(z_r)$ be the sets that are associated with the vertices z_1, z_2, \dots, z_r , respectively, by the FPP algorithm, when applied with canonical ordering σ_j . In order to prove that Γ and $\Phi = \Sigma^\ell$ are the same drawing and that $M_m^\pi(z_i) = M_m^\tau(z_i)$, for $i = 1, \dots, r$, it suffices to prove that:

- (i) for $j = \ell, \dots, m-1$, it holds that Σ^j and Σ^{j+1} are the same drawing and, for $j = \ell, \dots, m-1$ and for $i = 1, \dots, r$, it holds that $M_m^j(z_i) = M_m^{j+1}(z_i)$.
- (ii) Σ^m and Γ are the same drawing and, for $i = 1, \dots, r$, it holds that $M_m^m(z_i) = M_m^\pi(z_i)$.

We have that (ii) follows by Case 1. Indeed, Σ^m and Γ define the same canonical orientation of H , by Lemma 13, and the last vertex of both σ_m and π is $v_\ell = u_m$. Hence, it only remains to prove (i). Thus, consider any $j \in \{\ell, \dots, m-1\}$. We need to prove that Σ^j and Σ^{j+1} are the same drawing and that, for $i = 1, \dots, r$, it holds that $M_m^j(z_i) = M_m^{j+1}(z_i)$.

We introduce some notation. For $k = 3, \dots, m$, let H_k^j (let H_k^{j+1}) be the plane subgraph of H induced by the first k vertices of σ_j (resp. of σ_{j+1}). Also, let Σ_k^j (let Σ_k^{j+1}) be the drawing of H_k^j (resp. of H_k^{j+1}) that is constructed by the FPP algorithm when applied with canonical ordering σ_j (resp. σ_{j+1}), on the way of constructing Σ^j (resp. Σ^{j+1}). Furthermore, let $w_1^k = u, w_2^k, \dots, w_{r_k}^k = v$ (let $z_1^k = u, z_2^k, \dots, z_{s_k}^k = v$) be the clockwise order of the vertices along the outer face of H_k^j (resp. of H_k^{j+1}). Finally, for $i = 1, \dots, r_k$ (for $i = 1, \dots, s_k$), let $M_k^j(w_i^k)$ (resp. let $M_k^{j+1}(z_i^k)$) be the set that is associated to the vertex w_i^k (resp. to the vertex z_i^k) by the FPP algorithm, when applied with canonical ordering σ_j (resp. σ_{j+1}).

The whole reason for introducing the sequence of canonical orderings $\sigma_\ell, \sigma_{\ell+1}, \dots, \sigma_m$ of H is that canonical orderings that are consecutive in the sequence are very similar to one another. Indeed, σ_j and σ_{j+1} coincide, except for the j -th and $(j+1)$ -th vertex, which are swapped. Specifically, the j -th and $(j+1)$ -th vertex of σ_j are v_ℓ and v_{j+1} , respectively, while the j -th and $(j+1)$ -th vertex of σ_{j+1} are v_{j+1} and v_ℓ , respectively. This implies that, for $k = 3, \dots, j-1, j+1, \dots, m$, the graphs H_k^j and H_k^{j+1} coincide. Further, since σ_j and σ_{j+1} coincide on the first $j-1$ vertices, the drawings Σ_{j-1}^j and Σ_{j-1}^{j+1} of $H_{j-1}^j = H_{j-1}^{j+1}$ coincide and, for $i = 1, \dots, r_{j-1} = s_{j-1}$, the set $M_{j-1}^j(w_i^{j-1})$ coincides with $M_{j-1}^{j+1}(w_i^{j-1})$.

Note that H_j^j and H_j^{j+1} are not the same graph, as the vertex set of the former is $V(H_{j-1}^j) \cup \{v_\ell\}$, while the one of the latter is $V(H_{j-1}^j) \cup \{v_{j+1}\}$. Thus, obviously, Σ_j^j and Σ_j^{j+1} are not the same drawing, and the sets $M_j^j(w_i^j)$ and $M_j^{j+1}(z_i^j)$ associated to the vertices on the boundary of H_j^j and H_j^{j+1} , respectively, do not coincide. However, we are going to prove that all the required equalities are recovered at the next step of the FPP algorithm, when v_{j+1} and v_ℓ are inserted into H_j^j and H_j^{j+1} to form $H_{j+1}^j = H_{j+1}^{j+1}$.

Recall that the boundary of $H_{j-1}^j = H_{j-1}^{j+1}$ is $w_1^{j-1}, w_2^{j-1}, \dots, w_{r_{j-1}}^{j-1}$. In the discussion that follows, for ease of notation, we drop the apex $j-1$ from these vertices, which are then denoted by $w_1, w_2, \dots, w_{r_{j-1}}$. Since all the neighbors of v_ℓ precede v_ℓ in τ , by Property 5, and since v_{j+1} follows v_ℓ in τ , it follows that v_ℓ and v_{j+1} are not neighbors. This, together with Condition (CO-2) for H_j^j and H_{j+1}^j , implies that the neighbors of v_ℓ in $H_{j+1}^j = H_{j+1}^{j+1}$ are w_p, w_{p+1}, \dots, w_q , for some $1 \leq p < q \leq r_{j-1}$, that the neighbors of v_{j+1} in $H_{j+1}^j = H_{j+1}^{j+1}$ are $w_{p'}, w_{p'+1}, \dots, w_{q'}$, for some $1 \leq p' < q' \leq r_{j-1}$, and that either $q \leq p'$ or $q' \leq p$ (that is, the sequences of neighbors of v_ℓ and v_{j+1} along the boundary of $H_{j-1}^j = H_{j-1}^{j+1}$ are disjoint, except for the last vertex of one of them, which might coincide with the first vertex of the other one). Assume that $q \leq p'$, as the case $q' \leq p$ is symmetric. Then the graphs H_j^j , H_j^{j+1} , and $H_{j+1}^j = H_{j+1}^{j+1}$ have the following boundaries.

- H_j^j has boundary $w_1, \dots, w_p, v_\ell, w_q, \dots, w_{r_{j-1}}$;
- H_j^{j+1} has boundary $w_1, \dots, w_{p'}, v_{j+1}, w_{q'}, \dots, w_{r_{j-1}}$; and
- $H_{j+1}^j = H_{j+1}^{j+1}$ has boundary $w_1, \dots, w_p, v_\ell, w_q, \dots, w_{p'}, v_{j+1}, w_{q'}, \dots, w_{r_{j-1}}$.

Since $M_{j-1}^j(w_i) = M_{j-1}^{j+1}(w_i)$, for $i = 1, \dots, r_{j-1}$, the FPP algorithm defines:

- (i) $M_j^j(w_i) = M_{j-1}^j(w_i) \cup \{v_\ell\}$, for $i = 1, \dots, p$;
- (ii) $M_j^j(v_\ell) = M_{j-1}^j(w_{p+1}) \cup \{v_\ell\}$; and
- (iii) $M_j^j(w_i) = M_{j-1}^j(w_i)$, for $i = q, \dots, r_{j-1}$.
- (iv) $M_j^{j+1}(w_i) = M_{j-1}^j(w_i) \cup \{v_{j+1}\}$, for $i = 1, \dots, p'$;
- (v) $M_j^{j+1}(v_{j+1}) = M_{j-1}^j(w_{p'+1}) \cup \{v_{j+1}\}$; and
- (vi) $M_j^{j+1}(w_i) = M_{j-1}^j(w_i)$, for $i = q', \dots, r_{j-1}$.
- (vii) $M_{j+1}^j(w_i) = M_j^j(w_i) \cup \{v_{j+1}\}$, for $i = 1, \dots, p$;

- (viii) $M_{j+1}^j(v_\ell) = M_j^j(v_\ell) \cup \{v_{j+1}\};$
- (ix) $M_{j+1}^j(w_i) = M_j^j(w_i) \cup \{v_{j+1}\},$ for $i = q, \dots, p';$
- (x) $M_{j+1}^j(v_{j+1}) = M_j^j(w_{p'+1}) \cup \{v_{j+1}\};$ and
- (xi) $M_{j+1}^j(w_i) = M_j^j(w_i),$ for $i = q', \dots, r_{j-1}.$
- (xii) $M_{j+1}^{j+1}(w_i) = M_j^{j+1}(w_i) \cup \{v_\ell\},$ for $i = 1, \dots, p;$
- (xiii) $M_{j+1}^{j+1}(v_\ell) = M_j^{j+1}(w_{p+1}) \cup \{v_\ell\};$
- (xiv) $M_{j+1}^{j+1}(w_i) = M_j^{j+1}(w_i),$ for $i = q, \dots, p';$
- (xv) $M_{j+1}^{j+1}(v_{j+1}) = M_j^{j+1}(v_{j+1});$ and
- (xvi) $M_{j+1}^{j+1}(w_i) = M_j^{j+1}(w_i),$ for $i = q', \dots, r_{j-1}.$

We now prove that the required equalities for the sets associated by the FPP algorithm to the vertices along the boundary of $H_{j+1}^j = H_{j+1}^{j+1}$ are indeed satisfied.

- For $i = 1, \dots, p,$ by equalities (vii) and (i), we have $M_{j+1}^j(w_i) = M_{j-1}^j(w_i) \cup \{v_\ell, v_{j+1}\}.$ By equalities (xii) and (iv), we have $M_{j+1}^{j+1}(w_i) = M_{j-1}^j(w_i) \cup \{v_\ell, v_{j+1}\}.$ Hence, $M_{j+1}^j(w_i) = M_{j+1}^{j+1}(w_i),$ as required.
- By equalities (viii) and (ii), we have $M_{j+1}^j(v_\ell) = M_{j-1}^j(w_{p+1}) \cup \{v_\ell, v_{j+1}\}.$ By equalities (xiii) and (iv), we have $M_{j+1}^{j+1}(v_\ell) = M_{j-1}^j(w_{p+1}) \cup \{v_\ell, v_{j+1}\}.$ Hence, $M_{j+1}^j(v_\ell) = M_{j+1}^{j+1}(v_\ell),$ as required.
- For $i = q, \dots, p',$ by equalities (ix) and (iii), we have $M_{j+1}^j(w_i) = M_{j-1}^j(w_i) \cup \{v_{j+1}\}.$ By equalities (xiv) and (iv), we have $M_{j+1}^{j+1}(w_i) = M_{j-1}^j(w_i) \cup \{v_{j+1}\}.$ Hence, $M_{j+1}^j(w_i) = M_{j+1}^{j+1}(w_i),$ as required.
- By equalities (x) and (iii), we have $M_{j+1}^j(v_{j+1}) = M_{j-1}^j(w_{p'+1}) \cup \{v_{j+1}\}.$ By equalities (xv) and (v), we have $M_{j+1}^{j+1}(v_{j+1}) = M_{j-1}^j(w_{p'+1}) \cup \{v_{j+1}\}.$ Hence, $M_{j+1}^j(v_{j+1}) = M_{j+1}^{j+1}(v_{j+1}),$ as required.
- Finally, for $i = q' + 1, \dots, r_{j-1},$ by equalities (xi) and (iii), we have $M_{j+1}^j(w_i) = M_{j-1}^j(w_i).$ By equalities (xvi) and (vi), we have $M_{j+1}^{j+1}(w_i) = M_{j-1}^j(w_i).$ Hence, $M_{j+1}^j(w_i) = M_{j+1}^{j+1}(w_i),$ as required.

In order to prove that Σ_{j+1}^j and Σ_{j+1}^{j+1} are the same drawing, we first observe that all the vertices of H_{j-1}^j have in Σ_{j+1}^j and Σ_{j+1}^{j+1} the same y -coordinate as in Σ_{j-1}^j and in Σ_{j-1}^{j+1} , respectively. This is because the j -th and $(j+1)$ -th step of the FPP algorithm shift the vertices in the drawing of H_{j-1}^j only horizontally and because Σ_{j-1}^j and Σ_{j-1}^{j+1} are the same drawing.

In order to deal with the x -coordinates of the vertices of H_{j-1}^j in Σ_{j+1}^j and Σ_{j+1}^{j+1} , we partition the vertices of H_{j-1}^j into five sets V_0, V_1, V_2, V_3, V_4 defined as follows:

- $V_0 := M_{j-1}^j(w_1) \setminus M_{j-1}^j(w_{p+1})$,
- $V_1 := M_{j-1}^j(w_{p+1}) \setminus M_{j-1}^j(w_q)$,
- $V_2 := M_{j-1}^j(w_q) \setminus M_{j-1}^j(w_{p'+1})$,
- $V_3 := M_{j-1}^j(w_{p'+1}) \setminus M_{j-1}^j(w_{q'})$, and
- $V_4 := M_{j-1}^j(w_{q'})$.

We claim that, both in Σ_{j+1}^j and in Σ_{j+1}^{j+1} , for $i = 0, \dots, 4$, the x -coordinate of a vertex in V_i coincides with the one in $\Sigma_{j-1}^j = \Sigma_{j-1}^{j+1}$ plus i . This, together with the fact that Σ_{j-1}^j and Σ_{j-1}^{j+1} are the same drawing, implies that every vertex of H_{j-1}^j has the same x -coordinate in Σ_{j+1}^j and in Σ_{j+1}^{j+1} .

By the FPP algorithm, the x -coordinate of every vertex z of H_{j-1}^j in Σ_{j+1}^j is the same as in Σ_{j-1}^j , plus one unit for each set z belongs to among $M_{j-1}^j(w_{p+1})$, $M_{j-1}^j(w_q)$, $M_{j-1}^j(w_{p'+1})$, and $M_{j-1}^j(w_{q'})$. By equality (iii), the last two sets coincide with $M_{j-1}^j(w_{p'+1})$ and $M_{j-1}^j(w_{q'})$, respectively. Analogously, the x -coordinate of every vertex z in Σ_{j+1}^{j+1} is the same as in Σ_{j-1}^j , plus one unit for each set z belongs to among $M_{j-1}^{j+1}(w_{p'+1})$, $M_{j-1}^{j+1}(w_{q'})$, $M_{j-1}^{j+1}(w_{p+1})$, and $M_{j-1}^{j+1}(w_q)$. The first two sets coincide with $M_{j-1}^j(w_{p'+1})$ and $M_{j-1}^j(w_{q'})$, respectively. Further, by equality (iv), the last two sets coincide with $M_{j-1}^j(w_{p+1}) \cup \{v_{j+1}\}$ and $M_{j-1}^j(w_q) \cup \{v_{j+1}\}$, respectively.

- First, any vertex $z \in V_0$ belongs to neither of the sets $M_{j-1}^j(w_{p+1}) \cup \{v_{j+1}\}$, $M_{j-1}^j(w_q) \cup \{v_{j+1}\}$, $M_{j-1}^j(w_{p'+1})$, $M_{j-1}^j(w_{q'})$, given that v_{j+1} does not belong to H_{j-1}^j , given that $V_0 \cap M_{j-1}^j(w_{p+1}) = \emptyset$, by definition, and given that $M_{j-1}^j(w_{p+1}) \supseteq M_{j-1}^j(w_q) \supset M_{j-1}^j(w_{p'+1}) \supseteq M_{j-1}^j(w_{q'})$. Hence, the x -coordinate of z in both Σ_{j+1}^j and Σ_{j+1}^{j+1} coincides with the one in $\Sigma_{j-1}^j = \Sigma_{j-1}^{j+1}$.
- Second, any vertex $z \in V_1$ belongs to the set $M_{j-1}^j(w_{p+1})$, by definition, and to neither of the sets $M_{j-1}^j(w_q) \cup \{v_{j+1}\}$, $M_{j-1}^j(w_{p'+1})$, $M_{j-1}^j(w_{q'})$, given that v_{j+1} does not belong to H_{j-1}^j , given that $V_1 \cap M_{j-1}^j(w_q) = \emptyset$, by definition, and given that $M_{j-1}^j(w_q) \supset M_{j-1}^j(w_{p'+1}) \supseteq M_{j-1}^j(w_{q'})$. Hence, the x -coordinate of z in both Σ_{j+1}^j and Σ_{j+1}^{j+1} coincides with the one in $\Sigma_{j-1}^j = \Sigma_{j-1}^{j+1}$ plus one.
- Third, any vertex $z \in V_2$ belongs to the sets $M_{j-1}^j(w_{p+1})$ and $M_{j-1}^j(w_q)$, by definition and since $M_{j-1}^j(w_{p+1}) \supseteq M_{j-1}^j(w_q)$, and to neither of the sets $M_{j-1}^j(w_{p'+1})$ and $M_{j-1}^j(w_{q'})$, given that $V_2 \cap M_{j-1}^j(w_{p'+1}) = \emptyset$, by definition, and given that $M_{j-1}^j(w_{p'+1}) \supseteq M_{j-1}^j(w_{q'})$. Hence, the x -coordinate of z in both Σ_{j+1}^j and Σ_{j+1}^{j+1} coincides with the one in $\Sigma_{j-1}^j = \Sigma_{j-1}^{j+1}$ plus two.
- Fourth, any vertex $z \in V_3$ belongs to the sets $M_{j-1}^j(w_{p+1})$, $M_{j-1}^j(w_q)$, and $M_{j-1}^j(w_{p'+1})$, by definition and since $M_{j-1}^j(w_{p+1}) \supseteq M_{j-1}^j(w_q) \supset M_{j-1}^j(w_{p'+1})$, and does not belong to $M_{j-1}^j(w_{q'})$, given that $V_3 \cap M_{j-1}^j(w_{q'}) = \emptyset$, by definition. Hence, the x -coordinate of z in both Σ_{j+1}^j and Σ_{j+1}^{j+1} coincides with the one in $\Sigma_{j-1}^j = \Sigma_{j-1}^{j+1}$ plus three.

- Finally, any vertex $z \in V_4$ belongs to all of the sets $M_{j-1}^j(w_{p+1})$, $M_{j-1}^j(w_q)$, $M_{j-1}^j(w_{p'+1})$, and $M_{j-1}^j(w_{q'})$, by definition and since $M_{j-1}^j(w_{p+1}) \supseteq M_{j-1}^j(w_q) \supset M_{j-1}^j(w_{p'+1}) \supseteq M_{j-1}^j(w_{q'})$. Hence, the x -coordinate of z in both Σ_{j+1}^j and Σ_{j+1}^{j+1} coincides with the one in $\Sigma_{j-1}^j = \Sigma_{j-1}^{j+1}$ plus four.

It remains to prove that v_ℓ and v_{j+1} have the same coordinates in Σ_{j+1}^j and Σ_{j+1}^{j+1} . Denote by $\Omega(z)$ the position of a vertex z in a drawing Ω .

- By construction, $\Sigma_j^j(v_\ell)$ coincides with the intersection point of the line with slope $+1$ through $\Sigma_{j-1}^j(w_p)$ and the line with slope -1 through the point two units to the right of $\Sigma_{j-1}^j(w_q)$. Since v_ℓ belongs neither to $M_j^j(w_{p'+1})$ nor to $M_j^j(w_{q'})$, as by equality (iii) such sets coincide with $M_{j-1}^j(w_{p'+1})$ and $M_{j-1}^j(w_{q'})$, respectively, it follows that $\Sigma_{j+1}^j(v_\ell) = \Sigma_j^j(v_\ell)$.

Note that $\Sigma_j^{j+1}(w_p) = \Sigma_{j-1}^{j+1}(w_p) = \Sigma_{j-1}^j(w_p)$; indeed, the first equality holds true because w_p belongs neither to $M_{j-1}^{j+1}(w_{p'+1})$ nor to $M_{j-1}^{j+1}(w_{q'})$, and the second equality holds true since $\Sigma_{j-1}^j = \Sigma_{j-1}^{j+1}$. Analogously, $\Sigma_j^{j+1}(w_q) = \Sigma_{j-1}^{j+1}(w_q) = \Sigma_{j-1}^j(w_q)$. Hence, $\Sigma_{j+1}^{j+1}(v_\ell)$ coincides with the intersection point of the line with slope $+1$ through $\Sigma_{j-1}^j(w_p)$ and the line with slope -1 through the point two units to the right of $\Sigma_{j-1}^j(w_q)$, thus v_ℓ has the same coordinates in Σ_{j+1}^j and Σ_{j+1}^{j+1} .

- By construction and since $\Sigma_{j-1}^j = \Sigma_{j-1}^{j+1}$, we have that $\Sigma_{j+1}^{j+1}(v_{j+1})$ coincides with the intersection point p of the line with slope $+1$ through $\Sigma_{j-1}^j(w_{p'})$ and the line with slope -1 through the point two units to the right of $\Sigma_{j-1}^j(w_{q'})$. Since v_{j+1} belongs both to $M_j^{j+1}(w_{p+1})$ and to $M_j^{j+1}(w_q)$, as by equality (iv) such sets coincide with $M_{j-1}^j(w_{p+1}) \cup \{v_{j+1}\}$ and with $M_{j-1}^j(w_q) \cup \{v_{j+1}\}$, respectively, it follows that $\Sigma_{j+1}^{j+1}(v_{j+1})$ coincides with the point two units to the right of p .

Note that $\Sigma_j^j(w_{p'})$ coincides with the point two units to the right of $\Sigma_{j-1}^j(w_{p'})$. Indeed, $w_{p'}$ belongs both to $M_{j-1}^j(w_{p+1})$ and to $M_{j-1}^j(w_q)$. Analogously, $\Sigma_j^j(w_{q'})$ coincides with the point two units to the right of $\Sigma_{j-1}^j(w_{q'})$. Hence, $\Sigma_{j+1}^j(v_{j+1})$ coincides with the intersection point of the line with slope $+1$ through the point two units to the right of $\Sigma_{j-1}^j(w_{p'})$ and the line with slope -1 through the point two units to the right of $\Sigma_{j-1}^j(w_{q'})$, thus coincides with the point two units to the right of p . Hence, v_{j+1} has the same coordinates in Σ_{j+1}^j and Σ_{j+1}^{j+1} .

This concludes the proof that Σ_{j+1}^j and Σ_{j+1}^{j+1} are the same drawing.

Finally, since Σ_{j+1}^j and Σ_{j+1}^{j+1} are the same drawing, since the sets $M_{j+1}^j(w_i^{j+1})$ and $M_{j+1}^{j+1}(z_i^{j+1} = w_i^{j+1})$ coincide, for $i = 1, \dots, r_{j+1} = s_{j+1}$, and since σ_j and σ_{j+1} coincide on the last $m - (j + 1)$ vertices, it follows that Σ_k^j and Σ_k^{j+1} are the same drawing, for $k = j + 2, \dots, m$, and that the sets $M_k^j(w_i^k)$ and $M_k^{j+1}(z_i^k = w_i^k)$ coincide, for $k = j + 2, \dots, m$ and for $i = 1, \dots, r_k = s_k$. Since the drawings Σ_m^j and Σ_m^{j+1} coincide with Σ^j and Σ^{j+1} , respectively, it follows that Σ^j and Σ^{j+1} are the same drawing, as required. Also, since for $i = 1, \dots, r = r_m = s_m$, the set $M_m^j(z_i)$ coincides with $M_m^j(w_i^m)$ and the set $M_m^{j+1}(z_i)$ coincides with $M_m^{j+1}(z_i^m)$, it follows that $M_m^j(z_i) = M_m^{j+1}(z_i)$, as required. This completes the induction and hence the proof of Claim 2 and Property 3. It follows that the function f is surjective, which concludes the proof of Theorem 5.

Lemma 11 and Theorem 5 imply the following.

Lemma 14. *Let G be an n -vertex maximal plane graph and let (u, v, z) be the cycle delimiting its outer face, where u , v , and z appear in this counter-clockwise order along the cycle. There exists an algorithm with $\mathcal{O}(n)$ setup time and $\mathcal{O}(n)$ space usage that lists all canonical drawings of G with base edge (u, v) with $\mathcal{O}(n)$ delay.*

Theorem 6. *Let G be an n -vertex maximal plane (planar) graph. There exists an algorithm \mathcal{C}_1 (resp. \mathcal{C}_2) with $\mathcal{O}(n)$ setup time and $\mathcal{O}(n)$ space usage that lists all canonical drawings of G with $\mathcal{O}(n)$ delay.*

Proof: Algorithm \mathcal{C}_1 uses the algorithm for the proof of Lemma 14 applied three times, namely once for each choice of the base edge among the three edges incident to the outer face of the given maximal plane graph. Algorithm \mathcal{C}_2 uses algorithm \mathcal{C}_1 applied $4n - 8$ times, since there are $4n - 8$ maximal plane graphs which are isomorphic to a given maximal plane graph (see the proof of Theorem 3). Note that any two canonical drawings produced by different applications of algorithm \mathcal{C}_1 differ on the three vertices incident to the outer face, or on their coordinates in the drawing. \square

5 Enumeration of Schnyder Woods and Schnyder Drawings

In this section, we show how the enumeration algorithm for canonical orientations from Section 3 can be used in order to provide efficient algorithms for the enumeration of Schnyder woods and Schnyder drawings. We start with the following theorems.

Theorem 7. *Let G be an n -vertex maximal plane (planar) graph. There exists an algorithm \mathcal{M}_1 (resp. \mathcal{M}_2) with $\mathcal{O}(n)$ setup time and $\mathcal{O}(n)$ space usage that lists all Schnyder woods of G with $\mathcal{O}(n)$ delay.*

Proof: We first discuss algorithm \mathcal{M}_1 , hence let G be an n -vertex maximal plane graph. As proved by de Fraysseix and Ossona De Mendez [30, Theorem 3.3], there is a bijection between the Schnyder woods of G and the canonical orientations of G . Given a canonical orientation \mathcal{D} of G , the corresponding Schnyder wood $\mathcal{W} = (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$ can be obtained as follows (see also [20, 22, 34, 35, 38, 53, 64]). For every internal vertex w of G , let e_1, \dots, e_k be the counter-clockwise order of the incoming edges at w in \mathcal{D} , where $k \geq 2$. Assign e_1 with color 1 and orient it (in \mathcal{W}) so that it is outgoing at w ; also, assign e_k with color 2 and orient it (in \mathcal{W}) so that it is outgoing at w ; finally, assign e_2, \dots, e_{k-1} with color 3 and orient them (in \mathcal{W}) so that they are incoming at w . The construction of \mathcal{W} is completed by assigning all the edges that are incident to the sink of \mathcal{D} and that do not belong to the boundary of G with color 3 and orienting them (in \mathcal{W}) so that they are incoming at the sink of \mathcal{D} . Since the construction of \mathcal{W} from \mathcal{D} can be easily implemented in $\mathcal{O}(n)$ time and space, it descends from Theorem 3 that algorithm \mathcal{M}_1 satisfies the required properties.

Algorithm \mathcal{M}_2 uses algorithm \mathcal{M}_1 applied $4n - 8$ times, since there are $4n - 8$ maximal plane graphs isomorphic to a given maximal planar graph (see the proof of Theorem 3). Note that any two Schnyder woods produced by different applications of algorithm \mathcal{M}_1 differ on the triple of vertices that have no outgoing edge. \square

We now turn our attention to the enumeration of the planar straight-line drawings produced by the algorithm by Schnyder [64], known as *Schnyder drawings*. We start by describing such an algorithm, which takes as input (see Fig. 12a):

- an n -vertex maximal plane graph G , whose outer face is delimited by a cycle (u, v, z) , where u , v , and z appear in this counter-clockwise order along the outer face of G ; and

- a Schnyder wood $\mathcal{W} = (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$ of G .

For ease of notation, we let u_1 , u_2 , and u_3 be alternative labels for u , v , and z , respectively, so that \mathcal{T}_i contains u_i , for $i = 1, 2, 3$. For a cycle \mathcal{C} in G , let $\#_f(\mathcal{C})$ denote the number of internal faces of G in the interior of \mathcal{C} .

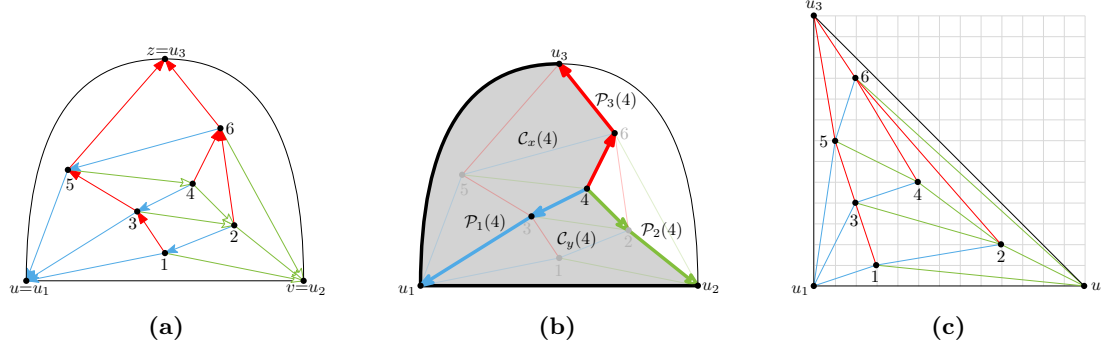


Figure 12: (a) A maximal plane graph G and a Schnyder wood \mathcal{W} of G . (b) Paths $\mathcal{P}_1(4)$, $\mathcal{P}_2(4)$, and $\mathcal{P}_3(4)$, and cycles $\mathcal{C}_x(4)$ and $\mathcal{C}_y(4)$, where 4 is an internal vertex of G . Note that $\#_f(\mathcal{C}_x(4)) = 5$ and $\#_f(\mathcal{C}_y(4)) = 5$. (c) The Schnyder drawing $s(\mathcal{W})$ of G .

Schnyder's algorithm assigns coordinates $(0, 0)$, $(2n - 5, 0)$, and $(0, 2n - 5)$ to the vertices u_1 , u_2 , and u_3 , respectively. Consider any internal vertex w . For $i = 1, 2, 3$, Properties (S-1) and (S-2) of \mathcal{W} imply that \mathcal{T}_i contains a directed path $\mathcal{P}_i(w)$ from w to u_i ; see Fig. 12b. Moreover, $\mathcal{P}_1(w)$, $\mathcal{P}_2(w)$, and $\mathcal{P}_3(w)$ have w as the only common vertex [64]. Let $\mathcal{C}_x(w)$ be the cycle composed of the paths $\mathcal{P}_1(w)$ and $\mathcal{P}_3(w)$, and of the edge (u_1, u_3) . Also, let $\mathcal{C}_y(w)$ be the cycle composed of the paths $\mathcal{P}_1(w)$ and $\mathcal{P}_2(w)$, and of the edge (u_1, u_2) . Then Schnyder's algorithm assigns coordinates $(\#_f(\mathcal{C}_x(w)), \#_f(\mathcal{C}_y(w)))$ to w ; see Fig. 12c.

We now prove the main ingredient of our enumeration algorithm for Schnyder drawings.

Theorem 8. *Let G be an n -vertex maximal plane graph. There exists a bijective function from the Schnyder woods of G to the Schnyder drawings of G . Also, given a Schnyder wood of G , the corresponding Schnyder drawing of G can be constructed in $\mathcal{O}(n)$ time.*

Proof: The bijective function s that proves the statement is simply Schnyder's algorithm. The second part of the statement then follows from the fact that this algorithm can be implemented in $\mathcal{O}(n)$ time [64].

In order to prove that s is bijective, we prove that it is injective (that is, for any two distinct Schnyder woods \mathcal{W}_1 and \mathcal{W}_2 of G , we have that $s(\mathcal{W}_1)$ and $s(\mathcal{W}_2)$ are not the same drawing) and that it is surjective (that is, for any Schnyder drawing Γ , there exists a Schnyder wood \mathcal{W} such that $s(\mathcal{W})$ is Γ). That s is surjective is actually obvious, as a Schnyder drawing Γ is generated by applying Schnyder's algorithm to some Schnyder's wood \mathcal{W} . Then $s(\mathcal{W}) = \Gamma$. In the following, we prove that s is injective.

Consider any Schnyder drawing Γ . By definition of Schnyder drawing, there is *at least* one Schnyder wood \mathcal{W} such that $s(\mathcal{W}) = \Gamma$. We prove that there is *at most* one such Schnyder wood, that is, Γ uniquely determines \mathcal{W} .

First, we recall that Schnyder drawings are often constructed on a triangular grid, rather than on the square grid. On the triangular grid, a Schnyder drawing constructed from a Schnyder wood

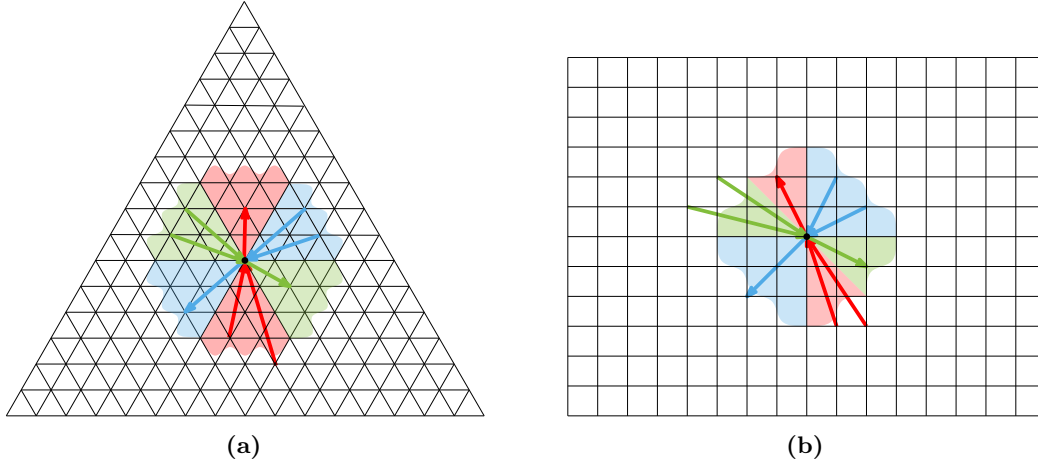


Figure 13: Slopes of the edges in a Schnyder drawing on (a) a triangular grid and (b) a square grid.

$\mathcal{W} = (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$ has the property that, for each vertex v , the edges of \mathcal{T}_1 , \mathcal{T}_2 , and \mathcal{T}_3 incoming into v have slopes in the intervals $(0^\circ, 60^\circ)$, $(120^\circ, 180^\circ)$, and $(240^\circ, 300^\circ)$, respectively, while the edges of \mathcal{T}_1 , \mathcal{T}_2 , and \mathcal{T}_3 outgoing from v have slopes in the intervals $(180^\circ, 240^\circ)$, $(300^\circ, 360^\circ)$, and $(60^\circ, 120^\circ)$, respectively; see, e.g., [35] and Fig. 13a. Hence, back on the square grid, a Schnyder drawing constructed from a Schnyder wood $\mathcal{W} = (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$ has the property that, for each vertex v , the edges of \mathcal{T}_1 , \mathcal{T}_2 , and \mathcal{T}_3 incoming into v have slopes in the intervals $(0^\circ, 90^\circ)$, $(135^\circ, 180^\circ)$, and $(270^\circ, 315^\circ)$, respectively, while the edges of \mathcal{T}_1 , \mathcal{T}_2 , and \mathcal{T}_3 outgoing from v have slopes in the intervals $(180^\circ, 270^\circ)$, $(315^\circ, 360^\circ)$, and $(90^\circ, 135^\circ)$, respectively; see Fig. 13b. Thus, for each edge (u, v) of G , whether (u, v) belongs to \mathcal{T}_1 , \mathcal{T}_2 , or \mathcal{T}_3 , and whether (u, v) is directed from u to v or vice versa, can be uniquely determined by the slope of the edge (u, v) in Γ . This concludes the proof that s is an injective function and hence the proof of Theorem 8. \square

We get the following.

Theorem 9. *Let G be an n -vertex maximal plane (planar) graph. There exists an algorithm \mathcal{N}_1 (resp. \mathcal{N}_2) with $\mathcal{O}(n)$ setup time and $\mathcal{O}(n)$ space usage that lists all Schnyder drawings of G with $\mathcal{O}(n)$ delay.*

Proof: Algorithm \mathcal{N}_1 directly descends from Theorem 7 and Theorem 8. Algorithm \mathcal{N}_2 uses algorithm \mathcal{N}_2 applied $4n - 8$ times, since there are $4n - 8$ maximal plane graphs which are isomorphic to a given maximal planar graph (see the proof of Theorem 3). Note that any two Schnyder drawings produced by different applications of algorithm \mathcal{N}_1 differ on the three vertices incident to the outer face, or on their coordinates in the drawing. \square

6 Comparison with Previous Work

In this section, we compare our results and algorithms with the literature.

First, and most notably, after the publication of the conference version of this paper [27], we became aware of an algorithm by Yamanaka and Nakano [72] for the enumeration of the so-called *left canonical orderings*. These are a specific type of canonical orderings, which the authors show to

be in bijection with the Schnyder woods. Thus, the algorithm by Yamanaka and Nakano can be exploited to enumerate all Schnyder woods of a maximal plane graph. Via the new bijections we proved in our paper, this can be employed to enumerate all drawings produced by the algorithms of de Fraysseix, Pach, and Pollack [33, 34] and of Schnyder [64]. The approach by Yamanaka and Nakano [72] is different from ours, as they work directly on the set of vertices that are candidate to be the vertex v_{k+1} following a partial canonical ordering v_1, \dots, v_k (see Condition (CO-2) for the properties that need to be satisfied by v_{k+1}), and filter out some candidate vertices in order to only obtain left canonical orderings. Conversely, we design an inductive approach that enumerates all the canonical orientations of our graph by modifying the canonical orientations of one or more smaller graphs. Our approach leads to confronting with topological structures and operations (including bipolar orientations, linear extensions of topological orderings, edge contractions and removals) which are interesting in their own right. Our algorithm for the enumeration of the canonical orientations of a maximal plane graph and of the drawings generated by these algorithms has been implemented and is publicly available [1].

Second, we remark that a different approach for the enumeration of canonical orientations might be based on the fact that the canonical orientations of a maximal plane graph form a distributive lattice \mathcal{L} [41]. By the fundamental theorem of finite distributive lattices [13], there is a finite poset P whose order ideals correspond to the elements of \mathcal{L} and it is known that $|P|$ is polynomial in n [41, page 10]. Enumerating the order ideals of P is a studied problem. In [44] an algorithm is presented that lists all order ideals of P in $\mathcal{O}(\Delta(P))$ delay, where $\Delta(P)$ is the maximum indegree of the covering graph of P . However, the algorithm has three drawbacks that make it unsuitable for solving our problems. First, the guaranteed delay of the algorithm is amortized, and not worst-case. Second, the algorithm uses $\mathcal{O}(w(P) \cdot |P|) = \mathcal{O}(n^3)$ space, where $w(P) = \mathcal{O}(n)$ is the width of P , and $\mathcal{O}(|P|^2) = \mathcal{O}(n^4)$ preprocessing time. Third and most importantly, each order ideal is produced twice by the algorithm, rather than just once as required by an enumeration algorithm. Similarly, the algorithms in [59, 67, 68] are affected by all or by part of the three drawbacks above.

Third, as mentioned in Section 1, our algorithm for enumerating all the canonical orientations of a maximal planar graph draws inspiration from the algorithms by de Fraysseix *et al.* [32] and by Setiawan and Nakano [65] for enumerating bipolar orientations of biconnected planar graphs. Note that the objective of our algorithm is different from the cited ones, as the canonical orientations form a subset of the bipolar orientations. Similarly to our algorithm, theirs also construct bipolar orientations inductively, by contracting and/or removing an edge, by listing the orientations of the resulting smaller graph, and by then suitably orienting the contracted and/or removed edge in order to obtain the orientations of the original graph. However, our algorithm has to deal with a different (and much more complicated) structure of the “partial orientations” (the so-called inner-canonical orientations) that allow to obtain canonical orientations of the entire graph. This led to the definition of well-formed graph, to the need of allowing multiple edges in certain parts of the graph, and to the need of removing many edges simultaneously.

7 Conclusions

In this paper, we considered the problem of enumerating two fundamental combinatorial structures of maximal planar graphs, i.e., canonical orderings and Schnyder woods. By exploiting their connection with canonical orientations, we developed efficient enumeration algorithms for such structures. We also proved novel, and in our opinion interesting, bijections between canonical orientations and canonical drawings, and between Schnyder woods and Schnyder drawings. This allowed us to empower our enumeration algorithms so that they can enumerate drawings within

these classical drawing styles. The worst-case delay between two consecutive outputs of all our algorithms is linear in the graph size.

Our research initiates the study of graph-drawing enumeration algorithms and sparkles several interesting questions in this domain. In general, given a graph G , for any given drawing style \mathcal{D} , we may ask for the existence of efficient algorithms to enumerate all drawings of G that respect \mathcal{D} . Natural examples of problems of this type include: (i) efficiently enumerating all the planar straight-line drawings of a given planar graph within a grid of prescribed size; (ii) efficiently enumerating all the orthogonal representations of a given plane graph with at most b bends in total; and (iii) efficiently enumerating all the upward planar embeddings of a single-source digraph or of a triconnected digraph.

Acknowledgments

We thank Vincenzo Politelli for contributing to the implementation [1] during his internship at Roma Tre University.

References

- [1] Source code for the algorithms presented in this paper. URL: <https://gitlab.com/uniroma3/computet/gd/projects/ice>.
- [2] Md. Jawaherul Alam, Therese Biedl, Stefan Felsner, Michael Kaufmann, Stephen G. Kobourov, and Torsten Ueckerdt. Computing cartograms with optimal complexity. *Discrete and Computational Geometry*, 50(3):784–810, 2013. doi:10.1007/s00454-013-9521-1.
- [3] Luca Castelli Aleardi and Olivier Devillers. SCARST: Schnyder compact and regularity sensitive triangulation data structure. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece*, volume 293 of *LIPICs*, pages 32:1–32:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: <https://doi.org/10.4230/LIPICs.SoCG.2024.32>, doi:10.4230/LIPICs.SoCG.2024.32.
- [4] Carlos Alegria, Susanna Caroppo, Giordano Da Lozzo, Marco D’Elia, Giuseppe Di Battista, Fabrizio Frati, Fabrizio Grosso, and Maurizio Patrignani. Upward pointset embeddings of planar st-graphs. In Stefan Felsner and Karsten Klein, editors, *32nd International Symposium on Graph Drawing and Network Visualization, GD 2024, September 18-20, 2024, Vienna, Austria*, volume 320 of *LIPICs*, pages 24:1–24:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: <https://doi.org/10.4230/LIPICs.GD.2024.24>, doi:10.4230/LIPICs.GD.2024.24.
- [5] Carlos Alegria, Susanna Caroppo, Giordano Da Lozzo, Marco D’Elia, Giuseppe Di Battista, Fabrizio Frati, Fabrizio Grosso, and Maurizio Patrignani. Upward pointset embeddings of planar st-graphs. *Algorithmica*, pages 1–31, 2025. doi:10.1007/s00453-025-01302-2.
- [6] Patrizio Angelini, Steven Chaplick, Sabine Cornelsen, Giordano Da Lozzo, and Vincenzo Roselli. Morphing triangle contact representations of triangulations. *Discrete and Computational Geometry*, 2023. doi:10.1007/s00454-022-00475-9.
- [7] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996. doi:10.1016/0166-218X(95)00026-N.

- [8] Imre Bárány and Günter Rote. Strictly convex drawings of planar graphs. *Documenta Mathematica*, 11:369–391, 2006. doi:[10.4171/DM/214](https://doi.org/10.4171/DM/214).
- [9] Jérémy Barbay, Luca Castelli Aleardi, Meng He, and J. Ian Munro. Succinct representation of labeled graphs. *Algorithmica*, 62(1-2):224–257, 2012. doi:[10.1007/s00453-010-9452-7](https://doi.org/10.1007/s00453-010-9452-7).
- [10] Valmir Carneiro Barbosa and Jayme Luiz Szwarcfiter. Generating all the acyclic orientations of an undirected graph. *Information Processing Letters*, 72(1-2):71–74, 1999. doi:[10.1016/S0020-0190\(99\)00120-9](https://doi.org/10.1016/S0020-0190(99)00120-9).
- [11] Ken Been, Eli Daiches, and Chee-Keng Yap. Dynamic map labeling. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):773–780, 2006. doi:[10.1109/TVCG.2006.136](https://doi.org/10.1109/TVCG.2006.136).
- [12] Therese C. Biedl, Joe Marks, Kathy Ryall, and Sue Whitesides. Graph multidrawing: Finding nice drawings without defining nice. In Sue Whitesides, editor, *Graph Drawing, 6th International Symposium, GD’98, Montréal, Canada, August 1998, Proceedings*, volume 1547 of *Lecture Notes in Computer Science*, pages 347–355. Springer, 1998. doi:[10.1007/3-540-37623-2_26](https://doi.org/10.1007/3-540-37623-2_26).
- [13] Garrett Birkhoff. Rings of sets. *Duke Mathematical Journal*, 3(3):443 – 454, 1937. doi:[10.1215/S0012-7094-37-00334-X](https://doi.org/10.1215/S0012-7094-37-00334-X).
- [14] Sarah Blind, Kolja Knauer, and Petru Valicov. Enumerating k-arc-connected orientations. *Algorithmica*, 82(12):3588–3603, 2020. doi:[10.1007/s00453-020-00738-y](https://doi.org/10.1007/s00453-020-00738-y).
- [15] A. Bondy and U.S.R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer London, 2011.
- [16] Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, Dominique Poulalhon, and Gilles Schaeffer. Planar graphs, via well-orderly maps and trees. *Graphs and Combinatorics*, 22(2):185–202, 2006. doi:[10.1007/s00373-006-0647-2](https://doi.org/10.1007/s00373-006-0647-2).
- [17] Prosenjit Bose, Vida Dujmovic, Ferran Hurtado, Stefan Langerman, Pat Morin, and David R. Wood. A polynomial bound for untangling geometric planar graphs. *Discrete and Computational Geometry*, 42(4):570–585, 2009. doi:[10.1007/s00454-008-9125-3](https://doi.org/10.1007/s00454-008-9125-3).
- [18] Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Competitive routing in the half- θ_6 -graph. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1319–1328. SIAM, 2012. doi:[10.1137/1.9781611973099.104](https://doi.org/10.1137/1.9781611973099.104).
- [19] Prosenjit Bose, Joachim Gudmundsson, and Michiel H. M. Smid. Constructing plane spanners of bounded degree and low weight. *Algorithmica*, 42(3-4):249–264, 2005. doi:[10.1007/s00453-005-1168-8](https://doi.org/10.1007/s00453-005-1168-8).
- [20] Enno Brehm. 3-orientations and Schnyder 3-tree-decompositions. Master’s thesis, Freie Universität Berlin, 2000.
- [21] Jason Cantarella, Robert B. Kusner, and John M. Sullivan. On the minimum ropelength of knots and links. *Inventiones Mathematicae*, 150:257–286, 2002. doi:[10.1007/s00222-002-0234-y](https://doi.org/10.1007/s00222-002-0234-y).
- [22] Luca Castelli Aleardi. *Algorithms for Graphs on Surfaces: From Graph Drawing to Graph Encoding*. Habilitation thesis, Université de Paris, 2021.

- [23] Marek Chrobak and Thomas H Payne. A linear-time algorithm for drawing a planar graph on a grid. *Information Processing Letters*, 54(4):241–246, 1995. doi:[10.1016/0020-0190\(95\)00020-D](https://doi.org/10.1016/0020-0190(95)00020-D).
- [24] Richie Chih-Nan Chuang, Ashim Garg, Xin He, Ming-Yang Kao, and Hsueh-I Lu. Compact encodings of planar graphs via canonical orderings and multiple parentheses. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *LNCS*, pages 118–129. Springer, 1998. doi:[10.1007/BFb0055046](https://doi.org/10.1007/BFb0055046).
- [25] Alessio Conte, Roberto Grossi, Andrea Marino, and Romeo Rizzi. Efficient enumeration of graph orientations with sources. *Discrete Applied Mathematics*, 246:22–37, 2018. doi:[10.1016/j.dam.2017.08.002](https://doi.org/10.1016/j.dam.2017.08.002).
- [26] Giordano Da Lozzo, Anthony D'Angelo, and Fabrizio Frati. On the area requirements of planar greedy drawings of triconnected planar graphs. In Donghyun Kim, R. N. Uma, Zhipeng Cai, and Dong Hoon Lee, editors, *26th International Conference on Computing and Combinatorics (COCOON '20)*, volume 12273 of *Lecture Notes in Computer Science*, pages 435–447. Springer, 2020. doi:[10.1007/978-3-030-58150-3_35](https://doi.org/10.1007/978-3-030-58150-3_35).
- [27] Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Fabrizio Grosso, and Maurizio Patrignani. Efficient enumeration of drawings and combinatorial structures for maximal planar graphs. In Ryuhei Uehara, Katsuhisa Yamanaka, and Hsu-Chun Yen, editors, *WALCOM: Algorithms and Computation*, pages 350–364, Singapore, 2024. Springer Nature Singapore. doi:[10.1007/978-981-97-0566-5_25](https://doi.org/10.1007/978-981-97-0566-5_25).
- [28] Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Upward planar morphs. In Therese Biedl and Andreas Kerren, editors, *International Symposium on Graph Drawing and Network Visualization*, pages 92–105. Springer, 2018. doi:[10.1007/978-3-030-04414-5_7](https://doi.org/10.1007/978-3-030-04414-5_7).
- [29] Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Upward planar morphs. *Algorithmica*, 82(10):2985–3017, 2020. doi:[10.1007/s00453-020-00714-6](https://doi.org/10.1007/s00453-020-00714-6).
- [30] Hubert de Fraysseix and Patrice Ossona de Mendez. On topological aspects of orientations. *Discrete Mathematics*, 229(1-3):57–72, 2001. URL: <https://www.sciencedirect.com/science/article/pii/S0012365X00002016>, doi:[10.1016/S0012-365X\(00\)00201-6](https://doi.org/10.1016/S0012-365X(00)00201-6).
- [31] Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. On triangle contact graphs. *Combinatorics, Probability and Computing*, 3:233–246, 1994. doi:[10.1017/S0963548300001139](https://doi.org/10.1017/S0963548300001139).
- [32] Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. Bipolar orientations revisited. *Discrete Applied Mathematics*, 56(2-3):157–179, 1995. doi:[10.1016/0166-218X\(94\)00085-R](https://doi.org/10.1016/0166-218X(94)00085-R).
- [33] Hubert de Fraysseix, János Pach, and Richard Pollack. Small sets supporting Fáry embeddings of planar graphs. In Janos Simon, editor, *20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pages 426–433. ACM, 1988. doi:[10.1145/62212.62254](https://doi.org/10.1145/62212.62254).

- [34] Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. doi:10.1007/BF02122694.
- [35] Raghavan Dhandapani. Greedy drawings of triangulations. *Discrete and Computational Geometry*, 43(2):375–392, 2010. doi:10.1007/s00454-009-9235-6.
- [36] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.
- [37] Giuseppe Di Battista, Fabrizio Grosso, Giulia Maragno, and Maurizio Patrignani. Ranking and unranking of the planar embeddings of a planar graph. In Shin-ichi Nakano and Mingyu Xiao, editors, *WALCOM: Algorithms and Computation - 19th International Conference and Workshops on Algorithms and Computation, WALCOM 2025, Chengdu, China, February 28 - March 2, 2025, Proceedings*, volume 15411 of *Lecture Notes in Computer Science*, pages 143–159. Springer, 2025. doi:10.1007/978-981-96-2845-2_10.
- [38] Giuseppe Di Battista, Roberto Tamassia, and Luca Vismara. Output-sensitive reporting of disjoint paths. *Algorithmica*, 23(4):302–340, 1999. doi:10.1007/PL00009264.
- [39] Vida Dujmovic, David Eppstein, Matthew Suderman, and David R. Wood. Drawings of planar graphs with few slopes and segments. *Computational Geometry*, 38(3):194–212, 2007. URL: <https://www.sciencedirect.com/science/article/pii/S0925772106000897>, doi:10.1016/j.comgeo.2006.09.002.
- [40] Stefan Felsner. Convex drawings of planar graphs and the order dimension of 3-polytopes. *Order*, 18(1):19–37, 2001. doi:10.1023/A:1010604726900.
- [41] Stefan Felsner. Lattice structures from planar graphs. *The Electronic Journal of Combinatorics*, 11(1), 2004. doi:10.37236/1768.
- [42] Stefan Felsner and Florian Zickfeld. Schnyder woods and orthogonal surfaces. *Discrete and Computational Geometry*, 40(1):103–126, 2008. doi:10.1007/s00454-007-9027-9.
- [43] Robert Ganian, Petr Hlinený, Joachim Kneis, Daniel Meister, Jan Obdržálek, Peter Rossmanith, and Somnath Sikdar. Are there any good digraph width measures? *Journal of Combinatorial Theory, Series B*, 116:250–286, 2016. URL: <https://www.sciencedirect.com/science/article/pii/S0095895615001082>, doi:10.1016/j.jctb.2015.09.001.
- [44] Michel Habib, Raoul Medina, Lhouari Nourine, and George Steiner. Efficient algorithms on distributive lattices. *Discrete Applied Mathematics*, 110(2-3):169–187, 2001. doi:10.1016/S0166-218X(00)00258-4.
- [45] Joel Hass and J. C. Lagarias. The number of Reidemeister moves needed for unknotting. *Journal of the American Mathematical Society*, 14:399–428, 2001. doi:10.1090/S0894-0347-01-00358-7.
- [46] Joel Hass, J. C. Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *Journal of the ACM*, 46(2):185–211, 1999. doi:10.1145/301970.301971.
- [47] Thomas P. Hayes. A simple condition implying rapid mixing of single-site dynamics on spin systems. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '06)*, pages 39–46. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.6.

- [48] Xin He, Ming-Yang Kao, and Hsueh-I Lu. Linear-time succinct encodings of planar graphs via canonical orderings. *SIAM Journal on Discrete Mathematics*, 12(3):317–325, 1999. doi:[10.1137/S0895480197325031](https://doi.org/10.1137/S0895480197325031).
- [49] Xin He and Huaming Zhang. A simple routing algorithm based on schnyder coordinates. *Theoretical Computer Science*, 494:112–121, 2013. Theory and Applications of Models of Computation (TAMC 2010). URL: <https://www.sciencedirect.com/science/article/pii/S0304397513000674>, doi:[10.1016/j.tcs.2013.01.017](https://doi.org/10.1016/j.tcs.2013.01.017).
- [50] Arthur B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962. doi:[10.1145/368996.369025](https://doi.org/10.1145/368996.369025).
- [51] Goos Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996. doi:[10.1007/BF02086606](https://doi.org/10.1007/BF02086606).
- [52] Donald E. Knuth. *The art of computer programming. Vol. 4A. Combinatorial algorithms. Part 1*. Addison-Wesley, 2011.
- [53] Stephen G. Kobourov. Canonical orders and Schnyder realizers. In *Encyclopedia of Algorithms*, pages 277–283. Springer, 2016. doi:[10.1007/978-1-4939-2864-4_650](https://doi.org/10.1007/978-1-4939-2864-4_650).
- [54] Kazuyuki Miura, Machiko Azuma, and Takao Nishizeki. Canonical decomposition, realizer, Schnyder labeling and orderly spanning trees of plane graphs. *International Journal of Foundations of Computer Science*, 16(1):117–141, 2005. doi:[10.1142/S0129054105002905](https://doi.org/10.1142/S0129054105002905).
- [55] Takao Nishizeki and Md. Saidur Rahman. *Planar Graph Drawing*, volume 12 of *Lecture Notes Series on Computing*. World Scientific, 2004. URL: <https://www.worldscientific.com/doi/abs/10.1142/5648>, doi:[10.1142/5648](https://doi.org/10.1142/5648).
- [56] Martin Nöllenburg, Roman Prutkin, and Ignaz Rutter. On self-approaching and increasing-chord drawings of 3-connected planar graphs. In Christian Duncan and Antonios Symvonis, editors, *Graph Drawing*, pages 476–487, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. doi:[10.1007/978-3-662-45803-7_40](https://doi.org/10.1007/978-3-662-45803-7_40).
- [57] Martin Nöllenburg, Roman Prutkin, and Ignaz Rutter. On self-approaching and increasing-chord drawings of 3-connected planar graphs. *Journal of Computational Geometry*, 7(1):47–69, 2016. doi:[10.20382/jocg.v7i1a3](https://doi.org/10.20382/jocg.v7i1a3).
- [58] Akimitsu Ono and Shin-Ichi Nakano. Constant time generation of linear extensions. In Maciej Liskiewicz and Rüdiger Reischuk, editors, *Fundamentals of Computation Theory, 15th International Symposium, FCT 2005, Lübeck, Germany, August 17-20, 2005, Proceedings*, volume 3623 of *Lecture Notes in Computer Science*, pages 445–453. Springer, 2005. doi:[10.1007/11537311_39](https://doi.org/10.1007/11537311_39).
- [59] Gara Pruesse and Frank Ruskey. Gray codes from antimatroids. *Order*, 10(3):239–252, 1993. doi:[10.1007/BF01110545](https://doi.org/10.1007/BF01110545).
- [60] Gara Pruesse and Frank Ruskey. Generating linear extensions fast. *SIAM Journal on Computing*, 23(2):373–386, 1994. doi:[10.1137/S0097539791202647](https://doi.org/10.1137/S0097539791202647).
- [61] Pierre Rosenstiehl and Robert Endre Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete and Computational Geometry*, 1:343–353, 1986. doi:[10.1007/BF02187706](https://doi.org/10.1007/BF02187706).

- [62] Frank Ruskey. *Combinatorial Generation*. University of Victoria, Victoria, BC, Canada, 2003.
- [63] Marcus Schaefer and Daniel Stefankovic. Decidability of string graphs. *Journal of Computer and System Sciences*, 68(2):319–334, 2004. URL: <https://www.sciencedirect.com/science/article/pii/S0022000003001375>, doi:10.1016/j.jcss.2003.07.002.
- [64] Walter Schnyder. Embedding planar graphs on the grid. In David S. Johnson, editor, *ACM-SIAM Symposium on Discrete Algorithms (SODA '90)*, pages 138–148. SIAM, 1990. URL: <http://dl.acm.org/citation.cfm?id=320176.320191>.
- [65] Andry Setiawan and Shin-ichi Nakano. Listing all st-orientations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 94(10):1965–1970, 2011. doi:10.1587/transfun.E94.A.1965.
- [66] Matthew B. Squire. Generating the acyclic orientations of a graph. *Journal of Algorithms*, 26(2):275–290, 1998. doi:10.1006/jagm.1997.0891.
- [67] Matthew Blaze Squire. *Gray Codes and Efficient Generation of Combinatorial Structures*. PhD thesis, North Carolina State University, 1995.
- [68] George Steiner. An algorithm to generate the ideals of a partial order. *Operations Research Letters*, 5(6):317–320, 1986. URL: <https://www.sciencedirect.com/science/article/pii/0167637786900714>, doi:10.1016/0167-6377(86)90071-4.
- [69] Roberto Tamassia, editor. *Handbook on Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013. URL: <https://cs.brown.edu/people/rtamassi/gdhandbook/>.
- [70] Kunihiro Wasa. Enumeration of enumeration algorithms. 2016. URL: <https://arxiv.org/abs/1605.05102>, arXiv:1605.05102.
- [71] Hassler Whitney. Non-separable and planar graphs. *Transactions of the American Mathematical Society*, 34(2):339–362, 1932. MR:1501641. Zbl:0004.13103. JFM:58.0608.01. doi:10.2307/1989545.
- [72] Katsuhisa Yamanaka and Shin-Ichi Nakano. Generating all realizers. *Electronics and Communications in Japan (Part II: Electronics)*, 89(7):40–47, 2006. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/ecjb.20276>, doi:10.1002/ecjb.20276.