# Efficient Enumeration of Drawings and Combinatorial Structures for Maximal Planar Graphs * 

Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Fabrizio Grosso, and Maurizio Patrignani<br>Roma Tre University, Rome, Italy<br>\{giordano.dalozzo, giuseppe.dibattista,fabrizio.frati,<br>fabrizio.grosso, maurizio.patrignani\}@uniroma3.it


#### 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.


## 1 Introduction

In the late eighties, de Fraysseix, Pach, and Pollack [25,26] and Schnyder [52] independently and almost simultaneously solved a question posed by Rosenstiehl and Tarjan [49] 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 [28], the result by de Fraysseix, Pach, and Pollack $[25,26]$ and by Schnyder [52] has a central place in the graph visualization literature. It also finds heterogeneous applications in other research areas, for example in knot theory $[15,36,37]$ and computational complexity $[7,34,51]$.

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., [44,57]. The combinatorial structures conceived for these algorithms have been used to solve a plethora of problems in graph drawing [1,2,4,20,23,27,30,31,33,41,45] and beyond $[5,11,12,13,18,38,39]$. 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 [52] 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 $\pi$ of

[^0]$G$, the canonical orientation of $G$ with respect to $\pi$ is the directed graph obtained from $G$ by orienting each edge away from the vertex that comes first in $\pi$. de Fraysseix and Ossona de Mendez [22] proved that there is a bijection between the canonical orientations and the Schnyder woods of $G$.

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., $[3,42,50,58]$. 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.
(ii) Machine-Learning-based graph drawing tools are eager of 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.

The enumeration of graph orientations has a rich literature. Consider an undirected graph $G$ with $n$ vertices and $m$ edges. In [19] 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 $\mathcal{O}(n m)$ delay; see also earlier works on the same problem [6,54]. In [9] the $k$-arc-connected orientations of $G$ are enumerated with $\mathcal{O}\left(\mathrm{knm}^{2}\right)$ 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 [24] provided an algorithm for enumerating the e-bipolar orientations of $G$ with polynomial delay. Setiawan and Nakano [53] 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 [22]: 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. [24] and enhanced by Setiawan and Nakano [53] 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 [22, 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 straightline 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.
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 $\mathcal{L}$ [32]. By the fundamental theorem of finite distributive lattices [8], 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 $n$ [32, page 10]. Enumerating the order ideals of $P$ is a studied problem. In [35] 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 $\mathcal{O}(w(P) \cdot|P|)=\mathcal{O}\left(n^{3}\right)$ space, where $w(P)=\mathcal{O}(n)$ is the width of $P$, and $\mathcal{O}\left(|P|^{2}\right)=\mathcal{O}\left(n^{4}\right)$ 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 $[47,55,56]$ are affected by all or by part of the three drawbacks above.

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). Conclusions and open problems are in Section 6.

## 2 Preliminaries

In the following, we provide basic definitions and concepts.
Graphs with multiple edges. For technical reasons, we consider graphs and digraphs with multiple 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 $\operatorname{deg}_{G}(v)$. Let $\mathcal{D}$ be a digraph. 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 $\Gamma$ 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 $\Gamma$ 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 $G^{\prime}$ of a plane graph $G$, we always assume that $G^{\prime}$ 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.

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 $e, e_{1}^{u}, \ldots, e_{h}^{u}$ and that the clockwise cyclic order of the edges incident to $v$ is $e, e_{1}^{v}, \ldots, e_{k}^{v}$. Then, the clockwise cyclic order of the edges incident to $w$ is set to $e_{1}^{u}, \ldots, e_{h}^{u}, e_{1}^{v}, \ldots, e_{k}^{v}$. Suppose that, in $G$, there exist a vertex $x$, an edge $e^{\prime}=(u, x)$, and an edge $e^{\prime \prime}=(v, x)$. Then, the contraction of $e$ in $G$ turns $e^{\prime}$ and $e^{\prime \prime}$ into a pair of parallel edges incident to $w$ and $x$. Also, suppose that there exists an edge $e^{\prime}$ that is parallel to $e$ in $G$. Then, the contraction of $e$ in $G$ turns $e^{\prime}$ 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 $G$ 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 $G_{1}, \ldots, G_{k}$ be the connected components of $G \backslash\{u, v\}$. The subgraphs of $G$ induced by $V\left(G_{i}\right) \cup\{u, v\}$, minus all parallel edges between $u$ and $v$, are components of $G$ separated by $\{u, v\}$, for $i=1, \ldots, k$. 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 [59]. 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 [10, 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.

Planar st-graphs. Given two vertices $s$ and $t$ of an undirected graph $G$, an orientation of $G$ is an storientation if (i) it is acyclic and (ii) $s$ and $t$ are its unique source and unique sink, respectively. A digraph is a planar $s t$-graph if and only if it is an $s t$-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 $s t$-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 $\operatorname{sink} t_{f}$. The following observations are well known.

Observation 1 ([21, 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 ([28, 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 $v$.

Let $\mathcal{D}$ be a plane $s t$-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 $v_{1}=u, v_{2}=v, v_{3}, \ldots, v_{n-1}, v_{n}=z$ meeting the following requirements for every $k=3, \ldots, n-1$; see Figs. 1a and 1 b and refer to [26].


Fig. 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 $G$.
(CO-1) The plane subgraph $G_{k} \subseteq G$ induced by $v_{1}, v_{2}, \ldots, v_{k}$ 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 $G^{\prime}$ is a maximal planar graph, a canonical ordering of $G^{\prime}$ is a canonical ordering of a maximal plane graph isomorphic to $G^{\prime}$.

Property 2. Let $\pi=\left(v_{1}=u, v_{2}=v, v_{3}, \ldots, v_{n}=z\right)$ be a canonical ordering of a maximal plane graph $G$. For $i=3, \ldots, n-1$, 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 $i=3, \ldots, n$, we denote by $G_{i}$ the plane subgraph of $G$ induced by $v_{1}, v_{2}, \ldots, v_{i}$. For $i=4, \ldots, n-1$, the fact that $v_{i}$ has at least two neighbors $v_{j}$ with $j<i$ directly follows by condition (CO-2) of $\pi$. Furthermore, $v_{3}$ is adjacent to $v_{1}$ and $v_{2}$, since $G_{3}$ is biconnected, by condition (CO-1) of $\pi$. Suppose, for a contradiction, that, for some $i \in\{3, \ldots, n-1\}$, the vertex $v_{i}$ has no neighbor $v_{j}$ with $j>i$. By condition (CO-1) of $\pi$, 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 $\pi$ 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 $\pi$.

Canonical orientations. Let $\pi=\left(v_{1}, \ldots, v_{n}\right)$ be a canonical ordering of $G$ with first vertex $u$. Orient every edge $\left(v_{i}, v_{j}\right)$ 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 $\pi$. We say that an orientation $\mathcal{D}$ of $G$ is a canonical orientation with first vertex $u$ if there exists a canonical ordering $\pi$ of $G$ with first vertex $u$ such that $\mathcal{D}$ is the canonical orientation of $G$ with respect to $\pi$; 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 $G^{\prime}$ is a maximal planar graph, a canonical orientation of $G^{\prime}$ is a canonical orientation of a maximal plane graph isomorphic to $G^{\prime}$.
Schnyder woods. A Schnyder wood $\left(\mathcal{T}_{1}, \mathcal{T}_{2}, \mathcal{T}_{3}\right)$ 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 [52]. 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 clockwise sector between the edges $e_{i+1}$ and $e_{i-1}$.


Fig. 2: 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 $C^{\prime}$ 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.
(S-2) At the outer vertices $u, v$, and $z$, all the internal edges are incoming and of color 1,2 , and 3 , respectively.

(S-1)

(S-2)

Finally, if $G^{\prime}$ is a maximal planar graph, a Schnyder wood of $G^{\prime}$ is a Schnyder wood of a maximal plane graph isomorphic to $G^{\prime}$.

## 3 Canonical Orientations

In [22, 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.

Theorem 1 ([22]). 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.

Proof: $(\Longrightarrow)$ Consider any canonical orientation $\mathcal{D}$ with first vertex $u$. We prove that $\mathcal{D}$ is a $u z$-orientation as in the statement. Let $\pi=\left(v_{1}=u, v_{2}=v, v_{3}, \ldots, v_{n-1}, v_{n}=z\right)$ be any canonical ordering of $G$ such that $\mathcal{D}$ is the canonical orientation of $G$ with respect to $\pi$. By the construction of $\mathcal{D}$ from $\pi$, an edge $\left(v_{i}, v_{j}\right)$ 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 $\operatorname{sink}$ in $\mathcal{D}$. Furthermore, $v_{2}=v$ has one incoming edge in $\mathcal{D}$, namely $\left(v_{1}, v_{2}\right)$, and at least one outgoing edge in $\mathcal{D}$, namely $\left(v_{2}, v_{3}\right)$. Finally, for $i=3,4, \ldots, n-1$, by Property 2 , we have that $v_{i}$ has at least two incoming edges and at least one outgoing edge in $\mathcal{D}$.
$(\Longleftarrow)$ Let $\mathcal{D}$ be a $u z$-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 $\pi=\left(v_{1}, \ldots, v_{n}\right)$ of
$\mathcal{D}$. We show that $\pi$ 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 $\mathcal{D}^{\prime}$ obtained from $\mathcal{D}$ by removing $u$ and $v$; refer to Fig. 2a. Consider any directed path $P$ from $w$ to $z$ in $\mathcal{D}^{\prime}$ 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 $C^{\prime}$ bounded by the edge $(u, w)$, by $P$, and by the edge $(u, z)$, or in the interior of the cycle $C^{\prime \prime}$ bounded by the edge $(v, w)$, by $P$, and by the edge $(v, z)$. In the former case, the edge $(v, s)$ crosses $C^{\prime}$, while in the latter case, the edge $(u, s)$ crosses $C^{\prime \prime}$. 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 $k=3, \ldots, n-1$, the ordering $\pi$ 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 $k \in\{3, \ldots, n-1\}$. Since $v_{k+1}$ has at least two incoming edges, by assumption, and since the end-vertices of such edges different from $v_{k+1}$ belong to $G_{k}$, since $\pi$ 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 $k=3, \ldots, n-1$, the vertex $v_{k+1}$ is in the outer face of $G_{k}$ (refer to Fig. 2b); 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 $\pi$ 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 $\left(w_{1}=u, w_{2}, \ldots, w_{m}=v\right)$ be such a path; refer to Fig. 2c. 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 $v_{k+2}, \ldots, v_{n}$ 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.

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 counterclockwise order along the cycle. By Theorem 1, we have that $\mathcal{D}$ is a $u z$-orientation in which every internal vertex has at least two incoming edges. The proof of Theorem 1 shows that every topological sorting of a $u z$-orientation in which every internal vertex has at least two incoming edges is a canonical ordering of $G$ with first vertex $u$.

Given two parallel edges $h_{1}$ and $h_{2}$ with end-vertices $s$ and $x$ in a plane graph, we denote by $\ell\left(h_{1}, h_{2}\right)$ the open region of the plane bounded by $h_{1}$ and $h_{2}$; we say that $\ell\left(h_{1}, h_{2}\right)$ is a multilens if it contains no vertices in its interior. Observe that $\ell\left(h_{1}, h_{2}\right)$ 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. 3, 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 $\ell\left(r_{1}, r_{2}\right), \ell\left(r_{2}, r_{3}\right)$, and $\ell\left(b_{2}, b_{3}\right)$ are also faces; loose edges are thick $\left(b_{3}\right.$ and $\left.r_{3}\right)$, whereas nonloose edges are thin $\left(b_{1}, b_{2}, r_{1}\right.$, and $\left.r_{2}\right)$.

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


Fig. 3: Illustration for the definition of a well-formed graph.

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. 3):

WF1: $s$ 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;
WF2: all the internal faces of $G$ have either two or three incident vertices;
WF3: multiple edges, if any, are all incident to $s$; and
WF4: if there exist two parallel edges $h_{1}$ and $h_{2}$ with end-vertices $s$ and $x$ such that $\ell\left(h_{1}, h_{2}\right)$ is not a multilens, then there exist two parallel edges $h_{1}^{\prime}$ and $h_{2}^{\prime}$ between $s$ and a vertex $y \neq x$ such that $\ell\left(h_{1}^{\prime}, h_{2}^{\prime}\right)$ is a multilens and such that $\ell\left(h_{1}^{\prime}, h_{2}^{\prime}\right) \subset \ell\left(h_{1}, h_{2}\right)$.

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 innercanonical 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 ( $w_{0}=s, w_{1}, \ldots, w_{k}=t$ ) be the right path $p_{r}$ of $G$. Let $e_{1}, e_{2}, \ldots, e_{m}$ 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 WF1. Let $v_{1}, \ldots, v_{m}$ be the end-vertices of $e_{1}, \ldots, e_{m}$ 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 $j \in\{1, \ldots, m-1\}$ 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 $e_{1}, \ldots, e_{j}$ 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 $\left\{s, w_{1}\right\}$ is a split pair of $G$. Let $G_{1}, G_{2}, \ldots, G_{k}$ be the components separated by $\left\{s, w_{1}\right\}$, where $k \geq 3$ and $G_{1}$ coincides with the edge $\left(s, w_{1}\right)$ of the right path of $G$. Since $G$ does not contain parallel edges, we have that $G_{2}, \ldots, G_{k}$ 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 WF2 of $G$. This proves that $G^{*}$ is biconnected.

We next prove that $G^{*}$ is well-formed; refer to Fig. 4.

- Condition WF1 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 $G$.


Fig. 4: Illustration for the contraction of the edge $e_{1}$.

- In order to prove Condition WF2, observe that $G^{*}$ contains no face incident to a single vertex, as the same is true for $G$, by Condition WF2 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 2 and that the contraction of an edge cannot increase the number of vertices incident to a face.
- Condition WF3 follows from the fact that $G$ satisfies Condition WF3 and that $G^{*}$ is obtained from $G$ by the contraction of an edge that has $s$ as an end-vertex; thus, new multiple edges, if any, are all incident to $s$.
- Finally, we prove Condition WF4. Consider any pair $\left(e_{1}, e_{2}\right)$ of parallel edges of $G^{*}$ such that $\ell\left(e_{1}, e_{2}\right)$ is not a multilens. By Condition WF3, 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 $\left(s, w_{1}\right)$ ), then there exist two parallel edges $e_{3}$ and $e_{4}$ such that $\ell\left(e_{3}, e_{4}\right)$ is a multilens and such that $\ell\left(e_{3}, e_{4}\right) \subset \ell\left(e_{1}, e_{2}\right)$ in $G^{*}$ as the same is true in $G$, given that $G$ satisfies Condition WF4. Otherwise, $e_{1}$ and $e_{2}$ are not parallel in $G$. Let $x_{1}, x_{2}, \ldots, x_{k}$ 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 $\left(s, w_{1}\right)$. Let $e^{\circ}=\left(s, x_{1}\right)$ and $e^{\diamond}=\left(w_{1}, x_{1}\right)$ be the edges incident to $f$ and different from $\left(s, w_{1}\right)$. Consider any pair $\left(e_{1}, e_{2}\right)$ of parallel edges of $G^{*}$ that are not parallel in $G$ and such that $\ell\left(e_{1}, e_{2}\right)$ is not a multilens. Then we have $e_{1}=\left(s, x_{i}\right)$ and $e_{2}=\left(s, x_{i}\right)$, with $2 \leq i \leq k$. However, in $G^{*}$, it holds that the edges $e^{\circ}$ and $e^{\diamond}$ define a multilens and that $\ell\left(e^{\circ}, e^{\diamond}\right) \subset \ell\left(e_{1}, e_{2}\right)$. Therefore Condition WF4 holds for $G^{*}$.

This concludes the proof that $G^{*}$ is well-formed, and the proof of the lemma.

Lemma 3. Suppose that $G$ contains parallel edges and let $j \in\{1, \ldots, m-1\}$ 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 $v_{2}, \ldots, v_{j}$ 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 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 $v_{2}, \ldots, v_{j}$ are not incident to the outer face of $G$; refer to Fig. 5. 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 $i=1, \ldots j$, the internal face of $G$ delimited by $e_{i}$ and $e_{i+1}$ is triangular. Consider the plane subgraph $P$ of $G$


Fig. 5: Illustration for the removal of the edges $e_{1}, \ldots, e_{j}$.
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 $a, b \in\{1, \ldots, j\}$ 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 WF4 (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 WF1 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 WF2 follows from the fact that $G$ satisfies Condition WF2 and that every internal face of $G^{-}$ is also an internal face of $G$, given that the edges $e_{1}, e_{2}, \ldots, e_{j}$ all lie outside $C_{o}^{-}$.
- Condition WF3 follows from the fact that $G$ satisfies Condition WF3 and that the edge set of $G^{-}$is a subset of the edge set of $G$.
- Finally, we prove Condition WF4. Consider any pair $\left(e_{p}, e_{q}\right)$ of parallel edges of $G^{-}$, where w.l.o.g. $p<q$, such that $\ell\left(e_{p}, e_{q}\right)$ is not a multilens. We prove that $\ell\left(e_{p}, e_{q}\right)$ 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 WF4, it contains two edges $e_{r}$ and $e_{s}$ such that $\ell\left(e_{r}, e_{s}\right)$ is a multilens and such that $\ell\left(e_{r}, e_{s}\right) \subset \ell\left(e_{p}, e_{q}\right)$, 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 $\ell\left(e_{p}, e_{q}\right)$ contains a multilens in its interior in $G^{-}$.

This concludes the proof that $G^{-}$is well-formed, and the proof of the lemma.
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 $\left(s, w_{1}\right)$ 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 $\left(s, w_{1}\right)$ 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 $\left(s, x_{1}\right)$ and $\left(s, w_{1}\right)$, which are both outgoing $s$ in $\mathcal{D}$, and by the edge $\left(w_{1}, x_{1}\right)$, which is outgoing $w_{1}$ in $\mathcal{D}$. Therefore $f$ is a $\diamond$-face with source $s$ and $\operatorname{sink} x_{1}$. The outer face of $G$ is bounded by the edge $(s, t)$ and the directed path $s, w_{1}, w_{2}, \ldots, t$, which are both outgoing $s$ in $\mathcal{D}$. Therefore the outer face is a $\diamond$-face with source $s$ and $\operatorname{sink} t$. 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 $\mathcal{D}^{*}$.

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 $e_{1}, e_{2}, \ldots, e_{j}$ 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 $s t$-orientation and that all the edges $e_{1}, e_{2}, \ldots, e_{j}$ 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 $i \in\{2, \ldots, j\}$. 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 $\mathcal{D}^{-}$.

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. 4. By Lemma 2, the plane graph $G^{*}$ obtained by the contraction of $e_{1}=\left(s, w_{1}\right)$ 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. 5a. 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 $e_{1}, e_{2}, \ldots, e_{j}$ is well-formed and thus, by induction, it admits an inner-canonical orientation $\mathcal{D}^{-}$. Also, by Lemma 5 , orienting the edges $e_{1}, e_{2}, \ldots, e_{j}$ 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 WF4, 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 WF4, 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 $\ell\left(e_{1}, e_{h}\right)$. 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 $G$.

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 graph with $\varphi$ edges. There exists an algorithm with $\mathcal{O}(\varphi)$ setup time and $\mathcal{O}(\varphi)$ space usage that lists all the inner-canonical orientations of $G$ with $\mathcal{O}(\varphi)$ 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 $u z$-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 $u z$-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 $u$.

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 $4 n-8$ times; this is because there are $4 n-8$ 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 $2 n-4$ 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.

### 3.1 The Inner-Canonical Enumerator Algorithm

We are now ready to describe an algorithm that takes in input a well-formed biconnected plane graph $G$ with poles $s$ and $t$, and enumerates all its inner-canonical orientations (confr. Theorem 2). The algorithm, which we call Inner-Canonical Enumerator (ICE, for short), works recursively; refer to Fig. 6. 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 $j \in\{2, \ldots, m-1\}$ 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 $i \in\{2, \ldots, j\}$ 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 $\left(s, w_{1}\right)$. 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 $\left(s, w_{1}\right)$ 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 $e_{1}, e_{2}, \ldots, e_{j}$. Let $G^{\prime}$ be the resulting plane graph and note that, by Lemma $3, G^{\prime}$ is biconnected and well-formed. Thus, the ICE algorithm can be applied recursively in order to enumerate all the inner-canonical orientations of $G^{\prime}$. The ICE algorithm then obtains all the inner-canonical orientations of $G$ as follows: For every inner-canonical orientation $\mathcal{D}^{\prime}$ of $G^{\prime}$, the algorithm constructs one inner-canonical orientation of $G$ by orienting the edges $e_{1}, e_{2}, \ldots, e_{j}$ away from $s$ and by keeping the orientation of all other edges unchanged.
- In Case 2, the ICE algorithm branches and applies both the contraction and the removal operations. More formally, first we contract the edge $\left(s, w_{1}\right)$, 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 $e_{1}, e_{2}, \ldots, e_{j}$ from $G$, obtaining a well-formed biconnected plane graph $G^{\prime}$, by Lemma 3. From every inner-canonical orientation $\mathcal{D}^{\prime}$ of $G^{\prime}$, 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 $\left(s, w_{1}\right)$ 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 $e_{1}, e_{2}, \ldots, e_{j}$ 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.

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 $\left(v_{1}, v_{2}\right)$ is outgoing $v_{1}$ in $\mathcal{D}$; recall that $e_{1}, \ldots, e_{m}$ 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 $v_{1}, \ldots, v_{m}$ are the end-vertices of $e_{1}, \ldots, e_{m}$ 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 $\left(v_{1}, v_{2}\right)$ 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 $e_{1}, e_{2}, \ldots, e_{j}$ 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 $e_{1}, e_{2}, \ldots e_{j}$ (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 $s t$-orientation of $G^{*}$. In the following, we first assume that the edge $\left(v_{1}, v_{2}\right)$ 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 $\left(v_{1}, v_{2}\right)$ is indeed outgoing $v_{1}$ in $\mathcal{D}$.

In view of Observation 1, in order to prove that $D^{*}$ is an $s t$-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 $\left(s, v_{1}, v_{2}\right)$. 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 $\left(v_{1}, v_{2}\right)$ 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 $\left(v_{1}, v_{2}\right)$ 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 $s, w_{2}, \ldots, t$, which are both outgoing $s$ in $\mathcal{D}^{*}$. Thus, the outer face of $\mathcal{D}^{*}$ is a $\diamond$-face with source $s$ and $\operatorname{sink} t$. This concludes the proof that $\mathcal{D}^{*}$ is an $s t$-orientation.

It remains to prove that the edge $\left(v_{1}, v_{2}\right)$ 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 $h \in\{1, \ldots, k\}$, we have that the internal faces of $G$ delimited by $e_{i}$ and $e_{i+1}$, for $i=1, \ldots, k-1$, 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 $v_{k}$ (confr. with 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 $v_{x}, v_{x-1}, \ldots, v_{1}$ of $P$ such that the edge $\left(v_{i}, v_{i-1}\right)$ is outgoing $v_{i}$, for $i=x, \ldots, 2$. 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 $\left(v_{x}, v_{x-1}\right)$ and $\left(v_{x}, v_{x+1}\right)$, that are both outgoing $v_{x}$ in $\mathcal{D}$. First, $\left(s, v_{x}\right)$ is incoming $v_{x}$ in $\mathcal{D}$ and it is the only edge incident to $v_{x}$ that follows $\left(v_{x}, v_{x+1}\right)$ and precedes $\left(v_{x}, v_{x-1}\right)$ in counter-clockwise order around $v_{x}$. Second, by Observation 2 , all the edges of $G$ incident to $v_{x}$ that follow $\left(v_{x}, v_{x-1}\right)$ and precede $\left(v_{x}, v_{x+1}\right)$ 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 $\left(v_{1}, v_{2}\right)$ 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 $s t$-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 $i=1, \ldots, j-1$, 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 $v_{j}$ (confr. with 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 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 $\left(v_{1}, v_{2}\right)$ is outgoing $v_{2}$. Then, we show that such an edge is outgoing $v_{2}$ in Case B . Consider the maximal directed subpath $v_{x}, \ldots, v_{1}$ of $P$ such the edge $\left(v_{i}, v_{i-1}\right)$ is outgoing $v_{i}$, for $i=x, \ldots, 2$. 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 $\left(v_{x}, v_{x-1}\right)$ and $\left(v_{x}, v_{x+1}\right)$ 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 $\left(v_{1}, v_{2}\right)$ is outgoing $v_{2}$.

Next, we prove that the edge $\left(v_{1}, v_{2}\right)$ 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 $\left(v_{2}, v_{1}\right)$ is outgoing $v_{2}$ since $\mathcal{D}$ is an $s t$-orientation. Otherwise, we have that the edge $\left(v_{1}=w_{1}, w_{2}\right)$ 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.

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 innercanonical 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 $e_{1}, e_{2}, \ldots, e_{j}$ 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 $e_{1}, e_{2}, \ldots, e_{j}$ 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.

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 $\left(v_{1}, v_{2}\right)$ 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 $\left(v_{1}, v_{2}\right)$ is outgoing $v_{1}$. On the other hand, in $\mathcal{D}_{2}^{-}$, the edge $\left(v_{1}, v_{2}\right)$ 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.

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

```
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 \(\leftarrow\) DetectCase()
    if case \(=\) CONTRACT or case \(=\) CONTRACT\&REMOVE then
            Edge \(e_{1} \leftarrow\) Contract()
            InnerCanonicalEnumerator ()
            Decontract ( \(e_{1}\) )
    if case \(=\) REMOVE or case \(=\) CONTRACT\&REMOVE then
            Edges[ ] removedEdges \(\leftarrow\) Remove()
            InnerCanonicalEnumerator ()
            Reinsert(removedEdges)
    if case \(=\) BASE then
            Output()
```


### 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 (confr. 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 $\operatorname{deg}_{G}(x)$;
- 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;

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;

- 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 $\varepsilon\left(e_{1}\right)$ 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 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 \(e_{1} \cdot x\) 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
```

- 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 $\varepsilon\left(e_{1}\right)$ 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.
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 $\mathcal{O}(\varphi)$ 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. 7. 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 $\varepsilon\left(e_{1}\right)$ 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 $\operatorname{deg}_{G^{*}}(s)$ of $s$ in $G^{*}$ to $\operatorname{deg}_{G}(s)+\operatorname{deg}_{G}\left(w_{1}\right)-2$. 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 $\left(w_{1}, w_{2}\right)$ 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 $\left(w_{1}, w_{2}\right)$. 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 $S$
 if $e$ is the $i$-th edge considered in the visit.


Fig. 7: 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.

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 $\nu(u)$ ). If that is the case and if $e \neq\left(w_{1}, w_{2}\right)$, 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 $\varepsilon\left(l_{c}\right)$ 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. 7, 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 $r_{\ell}$, 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 $\varepsilon\left(r_{\ell}\right)$.
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 $v_{2}, \ldots, v_{m}$; this information is stored in $\nu(x)$ as the reference to the rightmost edge $e_{j}$ between $x$ and $s$, for some $j \in\{2, \ldots, m\}$. 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 $\varepsilon\left(e_{j}\right)$. 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 $\varepsilon\left(l_{p}\right)$ 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 $\varepsilon\left(l_{c}\right)$ to the next chord in right-to-left order around $s$ to point to $\varepsilon\left(r_{c}\right)$, 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 $\varepsilon\left(l_{p}\right)$ to the next nonloose parallel edge in right-to-left order around $s$ to point to $\varepsilon\left(r_{p}\right)$, 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 be the successor of $e_{1}$ in $G$ in counter-clockwise order around $s$; refer to Fig. 7. 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 $\mathcal{O}\left(\operatorname{deg}_{G}\left(w_{1}\right)\right)$ time.
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 $\mathcal{O}\left(\operatorname{deg}_{G}\left(w_{1}\right)\right)$ time. A pseudocode description of the procedure is provided in Listing 5.
Remove. This procedure allows us to efficiently perform the removal of the edges $e_{1}, \ldots, e_{j}$ (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. 8. The procedure works as described next and returns an array REMOVED, whose entry with index $i$ points to the type-Edge record $\varepsilon\left(e_{i+1}\right)$ of the removed edge $e_{i+1}$, for $i=0, \ldots, j-1$.

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 $\varepsilon\left(e_{j}\right)$ 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 $\varepsilon\left(e_{j+1}\right)$ 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 $\varepsilon\left(e_{j}\right)$ 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. 8a), whereas it is different from $e_{j+1}$ if $e_{j}$ belongs to a multilens consisting of two parallel edges (see Fig. 8b).

```
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 \(\mathrm{e}_{1} \cdot \mathrm{x}\) is S then
    \(L e_{1}\).oriented_from_x_to_y \(\leftarrow\) True;
    else
    \(L e_{1}\).oriented_from_x_to_y \(\leftarrow\) False;
    In what follows, for simplicity, assume \(e_{1} \cdot x=s\) and \(e_{1} \cdot y=w_{1}\).
    \(S . e_{1} \leftarrow e_{1}\).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 \(w_{1}\) :
```



```
    \(e \leftarrow S . e_{1} ;\)
    for \(\mathrm{i} \leftarrow \mathbf{1}\); \(\mathrm{i}<\mathbf{e}_{\mathbf{1}}\).y.degree \(; \mathbf{i} \leftarrow \mathbf{i}+\mathbf{1}\) do
        Current edge \(e\) udpates (in what follows, for simplicity, assume \(e . y=w_{1}\) ):
        \(\overline{e . y \leftarrow S \text {; Identify } s \text { and } w_{1}}\).
        \(e\).ord \(\leftarrow S\).degree \(-i+1\)
        Handling new chords:
        if e.x.is_outer is True and e.isOuter is False then
            if \(l_{c}\) is NULL then
                    \(S . f i r s t\) _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 \(\leftarrow e\);
                \(e . n e x t \_n o n l o o s e \_l e n s ~ \leftarrow r_{\ell}\);
        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 \(\mathbf{i}<\mathrm{e}_{1} \cdot \mathrm{y}\).degree \(-\mathbf{1}\) 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 \(1_{c}\) is not NULL then
        \(\left\llcorner l_{c}\right.\).next_chord \(\leftarrow r_{c}\);
    if \(l_{p}\) is not NULL then
        \(\left\lfloor l_{p}\right.\).next_nonloose_parallel \(\leftarrow r_{p}\);
    e.next_around_y \(\leftarrow e_{1}\).next_around_x;
    return \(e_{1}\);
```

```
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 \(G\)
    \(e_{1}\).oriented_from_x_to_y \(\leftarrow\) NULL;
    \(S\).first_lens, that is \(h\), is not part of a multilens in \(G\).
    \(h \leftarrow S\).first_lens;
    \(b \leftarrow h\).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. \(x=s, e_{1} \cdot x=s\), and \(e_{1} \cdot y=w_{1}\).
        \(e . x \leftarrow e_{1} . y\);
        \(e . y\).first_incident_to_s \(\leftarrow e\).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.is_outer is False and e.y.is_outer is True then
            leftmost_chord \(\leftarrow e\);
        if \(e\).next_parallel_with_me is not NULL then
                leftmost_parallel \(\leftarrow e\);
        Current edge \(e\) is not incident to \(s\) in \(G\).
        \(e\).ord \(\leftarrow\) NULL;
        \(e\). next_chord \(\leftarrow\) NULL;
        \(e\).next_parallel_with_me \(\leftarrow\) NULL;
        \(e\).next_nonloose_parallel \(\leftarrow\) NULL;
        \(e\).next_nonloose_lens \(\leftarrow\) NULL;
        \(S\).degree \(\leftarrow S\).degree-1;
        \(e \leftarrow e\).next_around_x;
    \(h\).next_around_x \(\leftarrow e_{1}\);
    \(S\).first_lens \(\leftarrow h\).next_nonloose_lens;
    \(S\).first_chord \(\leftarrow\) leftmost_chord.next_chord;
    \(S\).first_parallel \(\leftarrow\) leftmost_parallel.next_nonloose_parallel;
    S. \(e_{1} \leftarrow e_{1}\);
```

(iii) The reference in $S$ to the rightmost parallel edge incident to $s$ is updated to the reference stored in $\varepsilon\left(e_{j}\right)$ 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. 8a and 8b.
(iv) The reference in $S$ to the rightmost edge incident to $s$ is updated to point to $\varepsilon\left(e_{j+1}\right)$.

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 $\varepsilon\left(e_{i}\right)$ 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 $\nu\left(v_{i}\right)$ representing the fact that this vertex is incident to the outer face of $G^{-}$to True, and the integer value in $\nu\left(v_{i}\right)$ representing the degree $\operatorname{deg}_{G^{-}}\left(v_{i}\right)$ of $v_{i}$ in $G^{-}$to $\operatorname{deg}_{G}\left(v_{i}\right)-1$. Furthermore, we add $e_{i}$ to the array REMOVED in the position with index $i-1$.
Updating new outer edges and outer vertices: For $h=1, \ldots, j-1$, denote by $e_{h, h+1}$ the edge $\left(v_{h}, v_{h+1}\right)$ and by $e_{0,1}$ the edge $\left(v_{1}=w_{1}, w_{2}\right)$ of $G$. We consider two cases based on whether $i \neq j$ or $i=j$.


Fig. 8: Illustrations for the Remove procedure. The input graph $G$ is on the left, whereas the graph $G^{-}$ resulting from the removal of $e_{1}, \ldots, e_{j}$ 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. 7. The vertices $v_{1}=w_{1}$ and $v_{3}$ that are neighbors of $s$ in $G$ but not in $G^{-}$are orange.

If $i \neq j$, then $e_{i}, e_{i+1}$, and $e_{i, i+1}$ bound a triangular face of $G$, and $\left(v_{i}, v_{i+1}\right)$ 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 $\varepsilon\left(e_{i, i+1}\right)$ 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 $\varepsilon\left(e_{i, i+1}\right)$ to the edge that follows it in counter-clockwise order around $v_{i}$ to point to $\varepsilon\left(e_{i-1, i}\right)$. Note that $\varepsilon\left(e_{i+1}\right)$ and $\varepsilon\left(e_{i-1, i}\right)$ can be accessed via $\varepsilon\left(e_{i}\right)$ as the record of the edge that follows $e_{i}$ in counter-clockwise order around $s$ and around $v_{i}$, respectively, and $\varepsilon\left(e_{i, i+1}\right)$ can be accessed via $\varepsilon\left(e_{i+1}\right)$ as the record of the edge that follows $e_{i+1}$ in counter-clockwise order around $v_{i+1}$.
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 $\varepsilon\left(e_{j+1}\right)$ that represents the fact that $e_{j+1}$ is incident to the outer face of $G^{-}$to True. Also, we set the reference in $\varepsilon\left(e_{j+1}\right)$ to the edge that follows it in counter-clockwise order around $v_{i}$ to point to $\varepsilon\left(e_{j-1, j}\right)$. Note that $\varepsilon\left(e_{j+1}\right)$ can be accessed via $\varepsilon\left(e_{i}\right)$ as above, and $\varepsilon\left(e_{j-1, j}\right)$ can be accessed via $\varepsilon\left(e_{j}\right)$ 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 $\left(s, v_{h}\right)$ 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 $e_{1}, \ldots, e_{j}$; refer to Fig. 8. If neither of $e_{1}, \ldots, e_{j-1}$ 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 $a^{\prime}$ and $b^{\prime}$ be the minimum and maximum indices of edges in $e_{1}, \ldots, e_{j}$ having parallel edges. Then the leftmost chord of $G^{-}$stemming from $e_{a^{\prime}}$ needs to be linked to the rightmost chord $r_{c}$ of $G$, whereas the rightmost chord stemming from $e_{b^{\prime}}$ 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 $\varepsilon\left(e_{i}\right)$ 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 $\nu\left(v_{i}\right)$ to the rightmost edge incident to $v_{i}$ that is also incident to $s$ to point to $\varepsilon\left(r_{p}^{i}\right)$. If $\varepsilon\left(r_{p}^{i}\right)$ 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 $\alpha$ 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 $\alpha$ in right-to-left order to point to the record of the edge $\beta$ parallel to $\alpha$ that follows $\alpha$ 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 $\varepsilon\left(l_{p}^{i}\right)$ 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 $e_{1}, \ldots, e_{i-1}$ and to the chords in $G$.

When all the edges $e_{1}, \ldots, e_{j}$ 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 $e_{1}, \ldots, e_{j-1}$ 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 $e_{1}, \ldots, e_{j}$ once and all the edges parallel to $e_{2}, \ldots, e_{j}$ 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 $\mathcal{O}\left(j+\sum_{i=2}^{j} \pi\left(e_{i}\right)\right)$, where $\pi\left(e_{i}\right)$ denotes the number of edges parallel to $e_{i}$ in $G$.
Reinsert. This procedure allows us to efficiently perform the "reinsertion" of the edges $e_{1}, \ldots, e_{j}$ 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 $\mathcal{O}\left(j+\sum_{i=2}^{j} \pi\left(e_{i}\right)\right)$. 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 $\varphi$ denotes the number of edges of $G$. Initializing the type-Vertex and the type-Edge records requires $\mathcal{O}(\varphi)$ 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 $\left(s, v_{1}\right)$ 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 $\left(s, v_{1}\right)$ 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 $\mathcal{O}(\varphi)$ time.
Space usage: At any step, the graph considered by the ICE algorithm has at most $\varphi$ edges. Thus, the space used to represent such a structure is $\mathcal{O}(\varphi)$. Also, the number of recursive calls to the ICE algorithm is $\mathcal{O}(\varphi)$. Therefore, in order to show that the overall space usage of the ICE algorithm is also $\mathcal{O}(\varphi)$, 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 $e_{1}, \ldots, e_{j}$. 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 $e_{1}, \ldots, e_{j}$, 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 $\mathcal{O}(\varphi)$.
Delay: We show that the time spent by the ICE algorithm to output the first inner-canonical orientation of $G$ is $\mathcal{O}(\varphi)$, and that the time between any two inner-canonical orientations of $G$ that are consecutively listed by the ICE algorithm is also $\mathcal{O}(\varphi)$. 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. 6. The root $\rho$ 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 $\nu$ be the parent node of a node $\mu$ of $\mathcal{T}$, and let $G_{\nu}$ and $G_{\mu}$ be the graphs associated with $\nu$ and $\mu$, respectively. The edge $(\nu, \mu)$ 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 $\lambda$ of $\mathcal{T}$, to the leaf of $\mathcal{T}$ that follows $\lambda$. 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 $e$.

```
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 \(e_{1}, \ldots, e_{j}\) of \(G\) removed as in Lemma 3.
    Side Effects: Sets the orientation of \(e_{1}, \ldots, e_{j}\) 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 \(S\) :
    \(\overline{S . d e g r e e} \leftarrow e_{j+1}\).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:
```



```
    while \(e\) is not \(e_{j+1}\) do
            Setting the orientation of \(e=\left(s, v_{i}\right)\) and updating the fields of \(\nu\left(v_{i}\right)\).
            e.oriented_from_x_to_y \(\leftarrow\) True; Orient \(e\)
            \(e . y\).isOuter \(\leftarrow\) True;
            \(e . y\).degree \(\leftarrow e . y\).degree -1 ;
            REMOVED[ \(e_{1}\).ord- \(e\). ord \(] \leftarrow \mathrm{e}\); Add \(e\) to the array of removed edges
            if \(e\) is not \(e_{j}\) then
                Updating the incidence list of \(v_{i}\) and making \(\left(v_{i}, v_{i+1}\right)\) an outer edge.
                    \(\overline{e_{i, i+1}} \leftarrow e\).next_around_x.next_around_y;
                    \(e_{i, i+1}\).isOuter \(\leftarrow\) 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.
                \(\overline{e_{j+1} \text {. isOuter } \leftarrow \text { True; }}\)
                \(e_{j+1}\).next_around_y \(\leftarrow e_{j}\).next_around_y;
            Updating chords:
            \(\overline{r_{p}^{i}} \leftarrow e\).next_parallel_with_me;
            \(e . y\).first_incident_to_s \(\leftarrow r_{p}^{i}\);
            if \(e=e_{j}\) 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 \(\left(s, v_{i}\right)\)
            \(\alpha \leftarrow r_{p}^{i}\);
            \(\beta \leftarrow \alpha\).next_parallel_with_me;
            while \(\beta\) 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 \(e_{1}, \ldots, e_{j}\) removed as in Lemma 3.
    Side Effects: Reinserts the edges \(e_{1}, \ldots, e_{j}\) in \(G^{-}\)to obtain \(G\).
    \(S . e_{1} \leftarrow \operatorname{REMOVED}[0] ;\) 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, \(e_{i} \cdot x=S, e_{i} . y=v_{i}\)
        if \(e_{i}\) is not \(e_{1}\) then
            \(L v_{i}\).isOuter \(\leftarrow\) 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 \(\left(v_{i}, v_{i+1}\right)\) an internal edge.
            We assume, for simplicity, \(e_{i, i+1}=\left(v_{i}, v_{i+1}\right), e_{i, i+1} \cdot x=v_{i}\), and \(e_{i, i+1} \cdot y=v_{i+1}\)
            \(e_{i+1} \leftarrow\) REMOVED[i];
            \(e_{i, i+1} \leftarrow e_{i+1}\).next_around_y;
            \(e_{i, i+1}\).next_around_x \(\leftarrow e_{i}\);
            \(e_{i, i+1}\).isOuter \(\leftarrow\) False;
        else
            Updating the incidence list of \(v_{j}\) and making \(e_{j+1}\) an internal edge.
            \(\overline{e_{j+1}} \leftarrow e_{j}\).next_around_x;
            \(e_{j+1}\).next_around_y \(\leftarrow e_{j}\);
            if \(e_{j+1}\).next_around_x is not NULL then
                \(e_{j+1}\).isOuter \(\leftarrow\) 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 \(\left(s, v_{i}\right)\).
                    \(e\). next_chord \(\leftarrow\) NULL;
                    \(e \leftarrow e\).next_parallel_with_me;
                new_first_chord \(\leftarrow e\).next_chord;
                \(e\).next_chord \(\leftarrow\) 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
$\bar{S}$.first_chord $\leftarrow$ new_first_chord;
$S$.first_parallel $\leftarrow$ new_first_parallel;
$S$.degree $\leftarrow S$.degree + REMOVED.length

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 $p_{\alpha}$ in $\mathcal{T}$ that connects $\rho$ and the first leaf $\alpha$ of $\mathcal{T}$, i.e., $\sum_{e \in p_{\alpha}} c(e)$, plus the time spent by the Output procedure. As the latter is $\mathcal{O}(\varphi)$, we only need to show that the former is also $\mathcal{O}(\varphi)$. 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 $k_{1} \cdot \operatorname{deg}_{G_{\nu}}\left(w_{1}\right)$, if $e$ corresponds to a Contract/DECONTRACT procedure, and is
at most $k_{2} \cdot\left(j+\sum_{i=2}^{j} \pi\left(e_{i}\right)\right)$, if $e$ corresponds to a REMOVE/REINSERT procedure that removes/reinsert the $j$ rightmost edges incident to the source of $G_{\nu}$, for suitable constants $k_{1}, k_{2}>0$. We have that $\sum_{e \in p_{\alpha}} c(e) \leq \max \left(k_{1}, k_{2}\right) \cdot 3 \varphi$. 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; an (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 $e_{2}, \ldots, e_{j}$, 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 $\lambda$ be a leaf of $\mathcal{T}$, let $\eta$ be the leaf of $\mathcal{T}$ that follows $\lambda$, and let $\xi$ be the lowest common ancestor of $\lambda$ and $\eta$. The time between the output of the inner-canonical orientation of $G$ corresponding to $\lambda$ and the output of the inner-canonical orientation of $G$ corresponding to $\eta$ coincides with the sum of the costs of all the edges of the path $q_{\lambda}$ in $\mathcal{T}$ between $\lambda$ and $\xi$ and of the costs of all the edges of the path $q_{\eta}$ in $\mathcal{T}$ between $\xi$ and $\eta$, 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 $\mathcal{O}(\varphi)$, we only need to show that the former is also $\mathcal{O}(\varphi)$. By the same arguments used above, we have that $\sum_{e \in q_{\lambda}} c(e)+\sum_{e \in q_{\eta}} c(e) \leq \max \left(k_{1}, k_{2}\right) \cdot 6 \varphi$, which is $O(\varphi)$.
This concludes the proof of Theorem 2.

## 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 $\pi$ of $G$ such that $\mathcal{D}$ is the canonical orientation of $G$ with respect to $\pi$ 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 $[46,48]$.

All generated canonical orderings have $u$ as first vertex since all the canonical orientations produced by the algorithms in $[46,48]$ 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 $[46,48]$ are different from one another. Moreover, any two canonical orderings $\pi$ and $\pi^{\prime}$ generated from different canonical orientations $\mathcal{D}$ and $\mathcal{D}^{\prime}$, respectively, differ as there exists an edge $\left(w, w^{\prime}\right)$ in $G$ which is oriented from $w$ to $w^{\prime}$ in $\mathcal{D}$ and from $w^{\prime}$ to $w$ in $\mathcal{D}^{\prime}$; this implies that $w$ precedes $w^{\prime}$ in $\pi$ and follows $w^{\prime}$ in $\pi^{\prime}$.

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 $4 n-8$ times, since there are $4 n-8$ maximal plane graphs


Fig. 9: Illustrations for the FPP algorithm.
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.

We now turn our attention to the enumeration of the planar straight-line drawings produced by the algorithm by de Fraysseix, Pach, and Pollack [26], 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 $\pi=\left(v_{1}=u, v_{2}=v, v_{3}, \ldots, v_{n}=z\right)$ of $G$; recall that $G_{k}$ denotes the subgraph of $G$ induced by the first $k$ vertices of $\pi$.

The FPP algorithm works in $n-2$ steps. At the first step, the FPP algorithm constructs a planar straight-line drawing $\Gamma_{3}$ 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 $M_{3}\left(v_{1}\right)=\left\{v_{1}, v_{2}, v_{3}\right\}, M_{3}\left(v_{3}\right)=\left\{v_{2}, v_{3}\right\}$, and $M_{3}\left(v_{2}\right)=\left\{v_{2}\right\}$.

For any $k=3, \ldots, n-1$, at step $k-1$, the FPP algorithm constructs a planar straight-line drawing $\Gamma_{k+1}$ of $G_{k+1}$ by suitably modifying $\Gamma_{k}$, as follows; refer to Fig. 9. Denote by $w_{1}=u, w_{2}, \ldots, w_{r}=v$ the clockwise order of the vertices of $G_{k}$ along its outer face. Assume that, during step $k-2$, the algorithm has defined, for $i=1, \ldots, r$, a subset $M_{k}\left(w_{i}\right)$ of the vertices of $G_{k}$ such that $M_{k}\left(w_{1}\right) \supset M_{k}\left(w_{2}\right) \supset \ldots \supset M_{k}\left(w_{r}\right)$. Let $w_{p}, w_{p+1}, \ldots, w_{q}$ be the neighbors of $v_{k+1}$ in $G_{k}$, for some $1 \leq p<q \leq r$. Then $\Gamma_{k+1}$ is obtained from $\Gamma_{k}$ by increasing the $x$-coordinate of each vertex in $M_{k}\left(w_{p+1}\right)$ by one unit, by increasing the $x$-coordinate of each vertex in $M_{k}\left(w_{q}\right)$ 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 $\Gamma_{k+1}$ is that the vertices $w_{1}, \ldots, w_{r}$ define in $\Gamma_{k}$ an $x$-monotone path whose edges have slope either +1 or -1 . The "shift" of the vertices in $M_{k}\left(w_{p+1}\right)$ makes room for drawing the edge $\left(w_{p}, v_{k+1}\right)$ with slope +1 , and the shift of the vertices in $M_{k}\left(w_{q}\right)$ makes room for drawing the edge $\left(w_{q}, v_{k+1}\right)$ with slope -1 , thus maintaining the invariant on the shape of the boundary of $\Gamma_{k+1}$. Step $k-1$ is completed by defining the sets:
$-M_{k+1}\left(w_{i}\right)=M_{k}\left(w_{i}\right) \cup\left\{v_{k+1}\right\}$, for $i=1, \ldots, p ;$
$-M_{k+1}\left(v_{k+1}\right)=M_{k}\left(w_{p+1}\right) \cup\left\{v_{k+1}\right\}$; and
$-M_{k+1}\left(w_{i}\right)=M_{k}\left(w_{i}\right)$, for $i=q, \ldots, r$.
We call canonical drawing with base edge $(u, v)$ the drawing $\Gamma_{n}$ 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 $\pi$, we say that $\pi$ extends $\mathcal{D}$ and that $\pi$ defines $\mathcal{D}$, depending on whether $\pi$ is constructed from $\mathcal{D}$ or vice versa. Also, we say that the canonical drawing $\Gamma$ of $G$ obtained by applying the FPP algorithm with a canonical ordering $\pi$ of $G$ corresponds to $\pi$.

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 $\pi$ 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 $\pi$. 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 $\pi$ ) 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 $\mathcal{D}$ [40], and that the FPP algorithm can be implemented in $\mathcal{O}(n)$ time [17].

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 $f\left(\mathcal{D}_{1}\right)$ and $f\left(\mathcal{D}_{2}\right)$ are not the same drawing) and that it is surjective (that is, for any canonical drawing $\Gamma$ with base edge $(u, v)$, there exists a canonical orientation $\mathcal{D}$ such that $f(\mathcal{D})$ is $\Gamma)$.

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 $\pi_{i}$ 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 $\pi_{1}$ and precedes $a$ in $\pi_{2}$. Hence, the $y$-coordinate of $b$ is larger than the one of $a$ in $f\left(\mathcal{D}_{1}\right)$ and smaller than the one of $a$ in $f\left(\mathcal{D}_{2}\right)$, thus $f\left(\mathcal{D}_{1}\right)$ and $f\left(\mathcal{D}_{2}\right)$ are not the same drawing.

We now prove that $f$ is surjective. Consider any canonical drawing $\Gamma$ of $G$ with base edge $(u, v)$ and let $\pi$ be a canonical ordering of $G$ with first vertex $u$ such that the canonical drawing corresponding to $\pi$ is $\Gamma$. Let $\mathcal{D}$ be the canonical orientation of $G$ with respect to $\pi$. 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 $\tau$, possibly different from $\pi$, that extends $\mathcal{D}$, hence $f(\mathcal{D})$ is the canonical drawing corresponding to $\tau$ and it is not guaranteed that $f(\mathcal{D})=\Gamma$. However, we have the following claim.
Claim 1 Any two canonical orderings $\pi$ and $\tau$ that extend $\mathcal{D}$ are such that the canonical drawings of $G$ corresponding to $\pi$ and $\tau$ are the same drawing.

Claim 1 implies that $f$ is surjective. Indeed, the function $f$ considers some canonical ordering $\tau$, possibly different from $\pi$, that extends $\mathcal{D}$; by Claim 1, the drawing corresponding to $\tau$ is the same drawing as the one corresponding to $\pi$, that is, $\Gamma$. 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 ( $v_{1}=u, v_{2}=v, v_{3}, \ldots, v_{m-1}, v_{m}$ ) such that, for $k=3, \ldots, m$, the plane subgraph $H_{k}$ of $H$ induced by $v_{1}, v_{2}, \ldots, v_{k}$ 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 $\pi$ 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 $\pi$. We say that $\pi$ extends and defines $\mathcal{D}_{H}$. A canonical drawing $\Gamma$ of $H$ with base edge $(u, v)$ is a drawing obtained by applying the FPP algorithm with a canonical ordering $\pi$ of $H$. We say that $\Gamma$ corresponds to $\pi$.

We now state the following claim, which is more general than, and hence implies, Claim 1.
Claim 2 Any two canonical orderings $\pi$ and $\tau$ that extend $\mathcal{D}_{H}$ are such that the canonical drawings of $H$ corresponding to $\pi$ and $\tau$ 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 $z_{1}, z_{2}, \ldots, z_{r}$ be the clockwise order of the vertices along the outer face of $H$. Furthermore, let $M_{m}^{\pi}\left(z_{1}\right), M_{m}^{\pi}\left(z_{2}\right), \ldots, M_{m}^{\pi}\left(z_{r}\right)$ (let $M_{m}^{\tau}\left(z_{1}\right), M_{m}^{\tau}\left(z_{2}\right), \ldots, M_{m}^{\tau}\left(z_{r}\right)$ ) be the sets that are associated to the vertices $z_{1}, z_{2}, \ldots, z_{r}$, respectively, by the FPP algorithm, when applied with canonical ordering $\pi$ (resp. $\tau$ ).

Property 3. For $i=1,2, \ldots, r$, the sets $M_{m}^{\pi}\left(z_{i}\right)$ and $M_{m}^{\tau}\left(z_{i}\right)$ coincide.
Let $\pi=\left(u_{1}=u, u_{2}=v, \ldots, u_{m}\right)$ and $\tau=\left(v_{1}=u, v_{2}=v, \ldots, v_{m}\right)$. Also, let $\Gamma$ and $\Phi$ be the canonical drawings of $H$ that correspond to $\pi$ and $\tau$, 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 $M_{3}^{\pi}\left(z_{1}\right)=M_{3}^{\tau}\left(z_{1}\right)=\left\{u_{1}, u_{2}, u_{3}\right\}$, $M_{3}^{\pi}\left(z_{2}\right)=M_{3}^{\tau}\left(z_{2}\right)=\left\{u_{2}, u_{3}\right\}$, and $M_{3}^{\pi}\left(z_{3}\right)=M_{3}^{\tau}\left(z_{3}\right)=\left\{u_{2}\right\}$.

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 $\ell$ be the index such that $v_{\ell}$ is the same vertex as $u_{m}$. We observe the following simple facts.

Property 4. We have $\ell \in\{4, \ldots, m\}$.
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 $u_{m}$.

Property 5. All the neighbors of $v_{\ell}$ in $H$ precede $v_{\ell}$ in $\tau$.
Proof: Since $u_{m}$ is the last vertex of $\pi$, all the neighbors of $u_{m}$ in $H$ precede $u_{m}$ in $\pi$, hence they also precede $v_{\ell}=u_{m}$ in $\tau$, as $\pi$ and $\tau$ define the same canonical orientation of $H$.

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 $w_{1}, w_{2}, \ldots, w_{s}$ be the clockwise order of the vertices along the outer face of $H$ and let $w_{p}, w_{p+1}, \ldots, w_{q}$ be the neighbors of $u_{m}$ in $H$, for some $1 \leq p<q \leq s$. Also, let $\lambda$ and $\xi$ be the vertex orderings of $L$ obtained from $\pi$ and $\tau$, respectively, by removing the vertex $u_{m}$. Finally, let $M_{m-1}^{\lambda}\left(w_{1}\right), M_{m-1}^{\lambda}\left(w_{2}\right), \ldots, M_{m-1}^{\lambda}\left(w_{s}\right)\left(\right.$ let $\left.M_{m-1}^{\xi}\left(w_{1}\right), M_{m-1}^{\xi}\left(w_{2}\right), \ldots, M_{m-1}^{\xi}\left(w_{s}\right)\right)$ be the sets that are associated to the vertices $w_{1}, w_{2}, \ldots, w_{s}$ by the FPP algorithm, when applied with canonical ordering $\lambda$ (resp. $\xi$ ). We next prove the following.

Lemma 12. $\lambda$ and $\xi$ are canonical orderings of $L$; furthermore, $\lambda$ and $\xi$ 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 $\pi$, which is biconnected by Condition (CO-1) of $\pi$. We show that $\lambda$ is a canonical ordering of $L$; the proof that $\xi$ is also a canonical ordering of $L$ is analogous. The first and second vertex of $\lambda$ are $u$ and $v$, since the same is true for $\pi$ and since $m>3$. Also, for $k=3, \ldots, m-2$, we have that conditions (CO-1) and (CO-2) hold for the subgraph $L_{k}$ of $L$ induced by the first $k$ vertices in $\lambda$, given that this coincides with the subgraph $H_{k}$ of $H$ induced by the first $k$ vertices in $\pi$ and given that $\pi$ is a canonical ordering of $H$. Finally, $\lambda$ and $\xi$ define the same canonical orientation of $L$, since the orientations of $L$ defined by $\lambda$ and $\xi$ both coincide with the orientation obtained from $\mathcal{D}_{H}$ by removing $u_{m}$ and its incident edges.

By Lemma 12, we have that $\lambda$ and $\xi$ are canonical orderings of $L$. Let $\Lambda$ and $\Xi$ be the canonical drawings of $L$ corresponding to $\lambda$ and $\xi$, respectively. By induction, $\Lambda$ and $\Xi$ are the same drawing and, for $i=1,2, \ldots, s$, we have $M_{m-1}^{\lambda}\left(w_{i}\right)=M_{m-1}^{\xi}\left(w_{i}\right)$.

The FPP algorithm constructs $\Gamma$ from $\Lambda$ by shifting each vertex in $M_{m-1}^{\lambda}\left(w_{p+1}\right)$ by one unit to the right, by shifting each vertex in $M_{m-1}^{\lambda}\left(w_{q}\right)$ 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

(a) The canonical ordering $\pi$ of $H$.

(b) The canonical ordering $\tau$ of $H$.

Fig. 10: The case in which $u_{m}$ and $v_{m}$ are not the same vertex. In this example, we have $\ell=7$. Then $\sigma_{7}=\left(v_{1}, v_{2}, v_{3}, v_{4}, v_{5}, v_{6}, v_{7}, v_{8}, v_{9}, v_{10}\right), \sigma_{8}=\left(v_{1}, v_{2}, v_{3}, v_{4}, v_{5}, v_{6}, v_{8}, v_{7}, v_{9}, v_{10}\right), \sigma_{9}=$ $\left(v_{1}, v_{2}, v_{3}, v_{4}, v_{5}, v_{6}, v_{8}, v_{9}, v_{7}, v_{10}\right)$, and $\sigma_{10}=\left(v_{1}, v_{2}, v_{3}, v_{4}, v_{5}, v_{6}, v_{8}, v_{9}, v_{10}, v_{7}\right)$.

FPP algorithm constructs $\Phi$ from $\Xi$ by shifting each vertex in $M_{m-1}^{\xi}\left(w_{p+1}\right)$ by one unit to the right, by shifting each vertex in $M_{m-1}^{\xi}\left(w_{q}\right)$ 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 $\Xi$ coincides with $\Lambda$, since $M_{m-1}^{\xi}\left(w_{p+1}\right)=M_{m-1}^{\lambda}\left(w_{p+1}\right)$, and since $M_{m-1}^{\xi}\left(w_{q}\right)=M_{m-1}^{\lambda}\left(w_{q}\right)$, we have that $\Phi$ and $\Gamma$ are the same drawing, as required.

By construction, the FPP algorithm defines: (i) $M_{m}^{\pi}\left(w_{i}\right)=M_{m-1}^{\lambda}\left(w_{i}\right) \cup\left\{u_{m}\right\}$, for $i=1, \ldots, p$; (ii) $M_{m}^{\pi}\left(u_{m}\right)=M_{m-1}^{\lambda}\left(w_{p+1}\right) \cup\left\{u_{m}\right\}$; (iii) $M_{m}^{\pi}\left(w_{i}\right)=M_{m-1}^{\lambda}\left(w_{i}\right)$, for $i=q, \ldots, s$; (iv) $M_{m}^{\tau}\left(w_{i}\right)=M_{m-1}^{\xi}\left(w_{i}\right) \cup$ $\left\{u_{m}\right\}$, for $i=1, \ldots, p ; \mathbf{( v )} M_{m}^{\tau}\left(u_{m}\right)=M_{m-1}^{\xi}\left(w_{p+1}\right) \cup\left\{u_{m}\right\}$; and ( $\left.\mathbf{v i}\right) M_{m}^{\tau}\left(w_{i}\right)=M_{m-1}^{\xi}\left(w_{i}\right)$, for $i=q, \ldots, s$. Since $M_{m-1}^{\xi}\left(w_{i}\right)=M_{m-1}^{\lambda}\left(w_{i}\right)$, for $i=1, \ldots, s$, it follows that $M_{m}^{\pi}\left(z_{i}\right)=M_{m}^{\tau}\left(z_{i}\right)$, for $i=1, \ldots, r$, 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 $v_{\ell}$, for some $\ell \in\{4, \ldots, m-1\}$. Our proof uses a sequence of canonical orderings $\sigma_{\ell+1}, \sigma_{\ell+2}, \ldots, \sigma_{m}$ of $H$, where for $j=\ell+1, \ldots, m$, the ordering $\sigma_{j}$ is defined as $\left(v_{1}, \ldots, v_{\ell-1}, v_{\ell+1}, \ldots, v_{j}, v_{\ell}, v_{j+1}, \ldots, v_{m}\right)$. That is, the order of the vertices of $H$ in $\sigma_{j}$ coincides with the one in $\tau$, except that $v_{\ell}$ is shifted to the $j$-th position. See Fig. 10 for an example. We first prove that $\sigma_{\ell+1}, \sigma_{\ell+2}, \ldots, \sigma_{m}$ are indeed canonical orderings of $H$.

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

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

- Condition (CO-1). For $k=3, \ldots, \ell-1, j, j+1, \ldots, m-1$, we have that $H_{k}^{j}$ is biconnected, because it coincides with $H_{k}^{\tau}$, and because $H_{k}^{\tau}$ satisfies condition (CO-1). We next prove that the graphs $H_{\ell}^{j}$, $H_{\ell+1}^{j}, \ldots, H_{j-1}^{j}$ are biconnected. The vertices $v_{\ell+1}, v_{\ell+2}, \ldots, v_{j}$ are not neighbors of $v_{\ell}$ in $H$, since by Property 5 all the neighbors of $v_{\ell}$ in $H$ precede $v_{\ell}$ in $\tau$ and hence are among $v_{1}, v_{2}, \ldots, v_{\ell-1}$. Also, for $k=\ell, \ell+1, \ldots, j-1$, we have that $H_{k}^{j}$ is the graph obtained from $H_{k+1}^{\tau}$ by removing $v_{\ell}$ and its incident edges. Hence, for $k=\ell-1, \ell, \ldots, j-2$, the $(k+1)$-th vertex of $\sigma_{j}$ has at least two neighbors in $H_{k}^{j}$, given that it coincides with the $(k+2)$-th vertex of $\tau$, which has at least two neighbors in $H_{k+1}^{\tau}$ since $H_{k+1}^{\tau}$ satisfies condition (CO-2), and given that it is not a neighbor of $v_{\ell}$. This, together with the fact that $H_{\ell-1}^{j}$ is biconnected, implies the biconnectivity of $H_{\ell}^{j}, H_{\ell+1}^{j}, \ldots, H_{j-1}^{j}$.
- Condition (CO-2). For $k=3, \ldots, \ell-2, j, j+1, \ldots, m-1$, the $(k+1)$-th vertex of $\sigma_{j}$ is in the outer face of $H_{k}^{j}$ because it is the same vertex as the $(k+1)$-th vertex of $\tau$, because $H_{k}^{j}$ coincides with $H_{k}^{\tau}$, and because the $(k+1)$-th vertex of $\tau$ is in the outer face of $H_{k}^{\tau}$, given that $H_{k}^{\tau}$ satisfies condition (CO-2). For $k=\ell-1, \ell, \ldots, j-2$, the $(k+1)$-th vertex of $\sigma_{j}$ is in the outer face of $H_{k}^{j}$ because it is the same vertex as the $(k+2)$-th vertex of $\tau$, because $H_{k}^{j}$ is a subgraph of $H_{k+1}^{\tau}$, and because the $(k+2)$-th vertex
of $\tau$ is in the outer face of $H_{k+1}^{\tau}$, given that $H_{k+1}^{\tau}$ satisfies condition (CO-2). Finally, the $j$-th vertex of $\sigma_{j}$, 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 $\pi$, given that $H_{m-1}$ satisfies condition (CO-2).

This proves that $\sigma_{j}$ is a canonical ordering. The fact that the canonical orientation $\mathcal{D}_{j}$ defined by $\sigma_{j}$ is $\mathcal{D}_{H}$ follows from the following facts: (i) the canonical orientation defined by $\tau$ is $\mathcal{D}_{H}$; (ii) the orderings $\tau$ and $\sigma_{j}$ coincide on the vertices of $H$ different from $v_{\ell}$ - this implies that the orientation of every edge that is not incident to $v_{\ell}$ is the same in $\mathcal{D}_{j}$ and in $\mathcal{D}_{H}$; and (iii) the vertex $v_{\ell}$ follows its neighbors in $H$ both in $\tau$ and in $\sigma_{j}$ - this implies that the orientation of every edge incident to $v_{\ell}$ is the same in $\mathcal{D}_{j}$ and in $\mathcal{D}_{H}$.

In order to simplify the description, let $\sigma_{\ell}$ coincide with $\tau$. For $j=\ell, \ldots, m$, let $\Sigma^{j}$ be the canonical drawing of $H$ that corresponds to $\sigma_{j}$. Also, let $M_{m}^{j}\left(z_{1}\right), M_{m}^{j}\left(z_{2}\right), \ldots, M_{m}^{j}\left(z_{r}\right)$ be the sets that are associated with the vertices $z_{1}, z_{2}, \ldots, z_{r}$, respectively, by the FPP algorithm, when applied with canonical ordering $\sigma_{j}$. In order to prove that $\Gamma$ and $\Phi=\Sigma^{\ell}$ are the same drawing and that $M_{m}^{\pi}\left(z_{i}\right)=M_{m}^{\tau}\left(z_{i}\right)$, for $i=1, \ldots, r$, it suffices to prove that:
(i) for $j=\ell, \ldots, m-1$, it holds that $\Sigma^{j}$ and $\Sigma^{j+1}$ are the same drawing and, for $j=\ell, \ldots, m-1$ and for $i=1, \ldots, r$, it holds that $M_{m}^{j}\left(z_{i}\right)=M_{m}^{j+1}\left(z_{i}\right)$.
(ii) $\Sigma^{m}$ and $\Gamma$ are the same drawing and, for $i=1, \ldots, r$, it holds that $M_{m}^{m}\left(z_{i}\right)=M_{m}^{\pi}\left(z_{i}\right)$.

We have that (ii) follows by Case 1 . Indeed, $\Sigma^{m}$ and $\Gamma$ define the same canonical orientation of $H$, by Lemma 13, and the last vertex of both $\sigma_{m}$ and $\pi$ is $v_{\ell}=u_{m}$. Hence, it only remains to prove (i). Thus, consider any $j \in\{\ell, \ldots, m-1\}$. We need to prove that $\Sigma^{j}$ and $\Sigma^{j+1}$ are the same drawing and that, for $i=1, \ldots, r$, it holds that $M_{m}^{j}\left(z_{i}\right)=M_{m}^{j+1}\left(z_{i}\right)$.

We introduce some notation. For $k=3, \ldots, m$, let $H_{k}^{j}$ (let $H_{k}^{j+1}$ ) be the plane subgraph of $H$ induced by the first $k$ vertices of $\sigma_{j}$ (resp. of $\sigma_{j+1}$ ). Also, let $\Sigma_{k}^{j}$ (let $\Sigma_{k}^{j+1}$ ) 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 $\sigma_{j}$ (resp. $\sigma_{j+1}$ ), on the way of constructing $\Sigma^{j}$ (resp. $\Sigma^{j+1}$ ). Furthermore, let $w_{1}^{k}=u, w_{2}^{k}, \ldots, w_{r_{k}}^{k}=v$ (let $z_{1}^{k}=u, z_{2}^{k}, \ldots, z_{s_{k}}^{k}=v$ ) be the clockwise order of the vertices along the outer face of $H_{k}^{j}$ (resp. of $H_{k}^{j+1}$ ). Finally, for $i=1, \ldots, r_{k}$ (for $i=1, \ldots, s_{k}$ ), let $M_{k}^{j}\left(w_{i}^{k}\right)$ (resp. let $M_{k}^{j+1}\left(z_{i}^{k}\right)$ ) 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 $\sigma_{j}$ (resp. $\sigma_{j+1}$ ).

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

Note that $H_{j}^{j}$ and $H_{j}^{j+1}$ are not the same graph, as the vertex set of the former is $V\left(H_{j-1}^{j}\right) \cup\left\{v_{\ell}\right\}$, while the one of the latter is $V\left(H_{j-1}^{j}\right) \cup\left\{v_{j+1}\right\}$. Thus, obviously, $\Sigma_{j}^{j}$ and $\Sigma_{j}^{j+1}$ are not the same drawing, and the sets $M_{j}^{j}\left(w_{i}^{j}\right)$ and $M_{j}^{j+1}\left(z_{i}^{j}\right)$ 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 $v_{\ell}$ 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 $w_{1}^{j-1}, w_{2}^{j-1}, \ldots, w_{r_{j-1}}^{j-1}$. In the discussion that follows, for ease of notation, we drop the apex ${ }^{j-1}$ from these vertices, which are then denoted by $w_{1}, w_{2}, \ldots, w_{r_{j-1}}$. Since all the neighbors of $v_{\ell}$ precede $v_{\ell}$ in $\tau$, by Property 5 , and since $v_{j+1}$ follows $v_{\ell}$ in $\tau$, it follows that $v_{\ell}$ 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 $v_{\ell}$ in $H_{j+1}^{j}=H_{j+1}^{j+1}$ are $w_{p}, w_{p+1}, \ldots, w_{q}$, 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 $w_{p^{\prime}}, w_{p^{\prime}+1}, \ldots, w_{q^{\prime}}$, for some $1 \leq p^{\prime}<q^{\prime} \leq r_{j-1}$, and that either $q \leq p^{\prime}$ or $q^{\prime} \leq p$ (that is, the sequences of neighbors of $v_{\ell}$ 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 $q \leq p^{\prime}$, as the case $q^{\prime} \leq p$ 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 $w_{1}, \ldots, w_{p}, v_{\ell}, w_{q}, \ldots, w_{r_{j-1}}$;
$-H_{j}^{j+1}$ has boundary $w_{1}, \ldots, w_{p^{\prime}}, v_{j+1}, w_{q^{\prime}}, \ldots, w_{r_{j-1}}$; and
$-H_{j+1}^{j}=H_{j+1}^{j+1}$ has boundary $w_{1}, \ldots, w_{p}, v_{\ell}, w_{q}, \ldots, w_{p^{\prime}}, v_{j+1}, w_{q^{\prime}}, \ldots, w_{r_{j-1}}$.
Since $M_{j-1}^{j}\left(w_{i}\right)=M_{j-1}^{j+1}\left(w_{i}\right)$, for $i=1, \ldots, r_{j-1}$, the FPP algorithm defines:
(i) $M_{j}^{j}\left(w_{i}\right)=M_{j-1}^{j}\left(w_{i}\right) \cup\left\{v_{\ell}\right\}$, for $i=1, \ldots, p$;
(ii) $M_{j}^{j}\left(v_{\ell}\right)=M_{j-1}^{j}\left(w_{p+1}\right) \cup\left\{v_{\ell}\right\}$; and
(iii) $M_{j}^{j}\left(w_{i}\right)=M_{j-1}^{j}\left(w_{i}\right)$, for $i=q, \ldots, r_{j-1}$.
(iv) $M_{j}^{j+1}\left(w_{i}\right)=M_{j-1}^{j}\left(w_{i}\right) \cup\left\{v_{j+1}\right\}$, for $i=1, \ldots, p^{\prime}$;
(v) $M_{j}^{j+1}\left(v_{j+1}\right)=M_{j-1}^{j}\left(w_{p^{\prime}+1}\right) \cup\left\{v_{j+1}\right\}$; and
(vi) $M_{j}^{j+1}\left(w_{i}\right)=M_{j-1}^{j}\left(w_{i}\right)$, for $i=q^{\prime}, \ldots, r_{j-1}$.
(vii) $M_{j+1}^{j}\left(w_{i}\right)=M_{j}^{j}\left(w_{i}\right) \cup\left\{v_{j+1}\right\}$, for $i=1, \ldots, p$;
(viii) $M_{j+1}^{j}\left(v_{\ell}\right)=M_{j}^{j}\left(v_{\ell}\right) \cup\left\{v_{j+1}\right\}$;
(ix) $M_{j+1}^{j}\left(w_{i}\right)=M_{j}^{j}\left(w_{i}\right) \cup\left\{v_{j+1}\right\}$, for $i=q, \ldots, p^{\prime}$;
(x) $M_{j+1}^{j}\left(v_{j+1}\right)=M_{j}^{j}\left(w_{p^{\prime}+1}\right) \cup\left\{v_{j+1}\right\}$; and
(xi) $M_{j+1}^{j}\left(w_{i}\right)=M_{j}^{j}\left(w_{i}\right)$, for $i=q^{\prime}, \ldots, r_{j-1}$.
(xii) $M_{j+1}^{j+1}\left(w_{i}\right)=M_{j}^{j+1}\left(w_{i}\right) \cup\left\{v_{\ell}\right\}$, for $i=1, \ldots, p$;
(xiii) $M_{j+1}^{j+1}\left(v_{\ell}\right)=M_{j}^{j+1}\left(w_{p+1}\right) \cup\left\{v_{\ell}\right\}$;
(xiv) $M_{j+1}^{j+1}\left(w_{i}\right)=M_{j}^{j+1}\left(w_{i}\right)$, for $i=q, \ldots, p^{\prime}$;
(xv) $M_{j+1}^{j+1}\left(v_{j+1}\right)=M_{j}^{j+1}\left(v_{j+1}\right)$; and
(xvi) $M_{j+1}^{j+1}\left(w_{i}\right)=M_{j}^{j+1}\left(w_{i}\right)$, for $i=q^{\prime}, \ldots, r_{j-1}$.

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

In order to prove that $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$ are the same drawing, we first observe that all the vertices of $H_{j-1}^{j}$ have in $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$ the same $y$-coordinate as in $\Sigma_{j-1}^{j}$ and in $\Sigma_{j-1}^{j+1}$, 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 $\Sigma_{j-1}^{j}$ and $\Sigma_{j-1}^{j+1}$ are the same drawing.

In order to deal with the $x$-coordinates of the vertices of $H_{j-1}^{j}$ in $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$, 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:
$-V_{0}:=M_{j-1}^{j}\left(w_{1}\right) \backslash M_{j-1}^{j}\left(w_{p+1}\right)$,
$-V_{1}:=M_{j-1}^{j}\left(w_{p+1}\right) \backslash M_{j-1}^{j}\left(w_{q}\right)$,
$-V_{2}:=M_{j-1}^{j}\left(w_{q}\right) \backslash M_{j-1}^{j}\left(w_{p^{\prime}+1}\right)$,
$-V_{3}:=M_{j-1}^{j}\left(w_{p^{\prime}+1}\right) \backslash M_{j-1}^{j}\left(w_{q^{\prime}}\right)$, and
$-V_{4}:=M_{j-1}^{j}\left(w_{q^{\prime}}\right)$.
We claim that, both in $\Sigma_{j+1}^{j}$ and in $\Sigma_{j+1}^{j+1}$, for $i=0, \ldots, 4$, 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 $\Sigma_{j-1}^{j}$ and $\Sigma_{j-1}^{j+1}$ are the same drawing, implies that every vertex of $H_{j-1}^{j}$ has the same $x$-coordinate in $\Sigma_{j+1}^{j}$ and in $\Sigma_{j+1}^{j+1}$.

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

- First, any vertex $z \in V_{0}$ belongs to neither of the sets $M_{j-1}^{j}\left(w_{p+1}\right) \cup\left\{v_{j+1}\right\}, M_{j-1}^{j}\left(w_{q}\right) \cup\left\{v_{j+1}\right\}$, $M_{j-1}^{j}\left(w_{p^{\prime}+1}\right), M_{j-1}^{j}\left(w_{q^{\prime}}\right)$, given that $v_{j+1}$ does not belong to $H_{j-1}^{j}$, given that $V_{0} \cap M_{j-1}^{j}\left(w_{p+1}\right)=\emptyset$, by definition, and given that $M_{j-1}^{j}\left(w_{p+1}\right) \supseteq M_{j-1}^{j}\left(w_{q}\right) \supset M_{j-1}^{j}\left(w_{p^{\prime}+1}\right) \supseteq M_{j-1}^{j}\left(w_{q^{\prime}}\right)$. Hence, the $x-$ coordinate of $z$ in both $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$ 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 $M_{j-1}^{j}\left(w_{p+1}\right)$, by definition, and to neither of the sets $M_{j-1}^{j}\left(w_{q}\right) \cup\left\{v_{j+1}\right\}, M_{j-1}^{j}\left(w_{p^{\prime}+1}\right), M_{j-1}^{j}\left(w_{q^{\prime}}\right)$, given that $v_{j+1}$ does not belong to $H_{j-1}^{j}$, given that $V_{1} \cap M_{j-1}^{j}\left(w_{q}\right)=\emptyset$, by definition, and given that $M_{j-1}^{j}\left(w_{q}\right) \supset M_{j-1}^{j}\left(w_{p^{\prime}+1}\right) \supseteq M_{j-1}^{j}\left(w_{q^{\prime}}\right)$. Hence, the $x$-coordinate of $z$ in both $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$ 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 $M_{j-1}^{j}\left(w_{p+1}\right)$ and $M_{j-1}^{j}\left(w_{q}\right)$, by definition and since $M_{j-1}^{j}\left(w_{p+1}\right) \supseteq M_{j-1}^{j}\left(w_{q}\right)$, and to neither of the sets $M_{j-1}^{j}\left(w_{p^{\prime}+1}\right)$ and $M_{j-1}^{j}\left(w_{q^{\prime}}\right)$, given that $V_{2} \cap$ $M_{j-1}^{j}\left(w_{p^{\prime}+1}\right)=\emptyset$, by definition, and given that $M_{j-1}^{j}\left(w_{p^{\prime}+1}\right) \supseteq M_{j-1}^{j}\left(w_{q^{\prime}}\right)$. Hence, the $x$-coordinate of $z$ in both $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$ 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 $M_{j-1}^{j}\left(w_{p+1}\right), M_{j-1}^{j}\left(w_{q}\right)$, and $M_{j-1}^{j}\left(w_{p^{\prime}+1}\right)$, by definition and since $M_{j-1}^{j}\left(w_{p+1}\right) \supseteq M_{j-1}^{j}\left(w_{q}\right) \supset M_{j-1}^{j}\left(w_{p^{\prime}+1}\right)$, and does not belong to $M_{j-1}^{j}\left(w_{q^{\prime}}\right)$, given that $V_{3} \cap M_{j-1}^{j}\left(w_{q^{\prime}}\right)=\emptyset$, by definition. Hence, the $x$-coordinate of $z$ in both $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$ 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 $M_{j-1}^{j}\left(w_{p+1}\right), M_{j-1}^{j}\left(w_{q}\right), M_{j-1}^{j}\left(w_{p^{\prime}+1}\right)$, and $M_{j-1}^{j}\left(w_{q^{\prime}}\right)$, by definition and since $M_{j-1}^{j}\left(w_{p+1}\right) \supseteq M_{j-1}^{j}\left(w_{q}\right) \supset M_{j-1}^{j}\left(w_{p^{\prime}+1}\right) \supseteq M_{j-1}^{j}\left(w_{q^{\prime}}\right)$. Hence, the $x$-coordinate of $z$ in both $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$ coincides with the one in $\Sigma_{j-1}^{j}=\Sigma_{j-1}^{j+1}$ plus four.

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

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

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

- By construction and since $\Sigma_{j-1}^{j}=\Sigma_{j-1}^{j+1}$, we have that $\Sigma_{j}^{j+1}\left(v_{j+1}\right)$ coincides with the intersection point $p$ of the line with slope +1 through $\Sigma_{j-1}^{j}\left(w_{p^{\prime}}\right)$ and the line with slope -1 through the point two units to the right of $\Sigma_{j-1}^{j}\left(w_{q^{\prime}}\right)$. Since $v_{j+1}$ belongs both to $M_{j}^{j+1}\left(w_{p+1}\right)$ and to $M_{j}^{j+1}\left(w_{q}\right)$, as by equality (iv) such sets coincide with $M_{j-1}^{j}\left(w_{p+1}\right) \cup\left\{v_{j+1}\right\}$ and with $M_{j-1}^{j}\left(w_{q}\right) \cup\left\{v_{j+1}\right\}$, respectively, it follows that $\Sigma_{j+1}^{j+1}\left(v_{j+1}\right)$ coincides with the point two units to the right of $p$.
Note that $\Sigma_{j}^{j}\left(w_{p^{\prime}}\right)$ coincides with the point two units to the right of $\Sigma_{j-1}^{j}\left(w_{p^{\prime}}\right)$. Indeed, $w_{p^{\prime}}$ belongs both to $M_{j-1}^{j}\left(w_{p+1}\right)$ and to $M_{j-1}^{j}\left(w_{q}\right)$. Analogously, $\Sigma_{j}^{j}\left(w_{q^{\prime}}\right)$ coincides with the point two units to the right of $\Sigma_{j-1}^{j}\left(w_{q^{\prime}}\right)$. Hence, $\Sigma_{j+1}^{j}\left(v_{j+1}\right)$ coincides with the intersection point of the line with slope +1 through the point two units to the right of $\Sigma_{j-1}^{j}\left(w_{p^{\prime}}\right)$ and the line with slope -1 through the point two units to the right of $\Sigma_{j-1}^{j}\left(w_{q^{\prime}}\right)$, thus coincides with the point two units to the right of $p$. Hence, $v_{j+1}$ has the same coordinates in $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$.

This concludes the proof that $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$ are the same drawing.
Finally, since $\Sigma_{j+1}^{j}$ and $\Sigma_{j+1}^{j+1}$ are the same drawing, since the sets $M_{j+1}^{j}\left(w_{i}^{j+1}\right)$ and $M_{j+1}^{j+1}\left(z_{i}^{j+1}=w_{i}^{j+1}\right)$ coincide, for $i=1, \ldots, r_{j+1}=s_{j+1}$, and since $\sigma_{j}$ and $\sigma_{j+1}$ coincide on the last $m-(j+1)$ vertices, it follows that $\Sigma_{k}^{j}$ and $\Sigma_{k}^{j+1}$ are the same drawing, for $k=j+2, \ldots, m$, and that the sets $M_{k}^{j}\left(w_{i}^{k}\right)$ and $M_{k}^{j+1}\left(z_{i}^{k}=w_{i}^{k}\right)$ coincide, for $k=j+2, \ldots, m$ and for $i=1, \ldots, r_{k}=s_{k}$. Since the drawings $\Sigma_{m}^{j}$ and $\Sigma_{m}^{j+1}$ coincide with $\Sigma^{j}$ and $\Sigma^{j+1}$, respectively, it follows that $\Sigma^{j}$ and $\Sigma^{j+1}$ are the same drawing, as required. Also, since for $i=1, \ldots, r=r_{m}=s_{m}$, the set $M_{m}^{j}\left(z_{i}\right)$ coincides with $M_{m}^{j}\left(w_{i}^{m}\right)$ and the set $M_{m}^{j+1}\left(z_{i}\right)$ coincides with $M_{m}^{j+1}\left(z_{i}^{m}\right)$, it follows that $M_{m}^{j}\left(z_{i}\right)=M_{m}^{j+1}\left(z_{i}\right)$, 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 $4 n-8$ times, since there are $4 n-8$ 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.

## 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 [22, 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 $\mathcal{W}=\left(\mathcal{T}_{1}, \mathcal{T}_{2}, \mathcal{T}_{3}\right)$ can be obtained as follows (see also [14,16,26,27,29,43,52]). For every internal vertex $w$ of $G$, let $e_{1}, \ldots, e_{k}$ 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 $e_{2}, \ldots, e_{k-1}$ 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 $4 n-8$ times, since there are $4 n-8$ 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.

We now turn our attention to the enumeration of the planar straight-line drawings produced by the algorithm by Schnyder [52], known as Schnyder drawings. We start by describing such an algorithm, which takes as input (see Fig. 11a):

- 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 $\mathcal{W}=\left(\mathcal{T}_{1}, \mathcal{T}_{2}, \mathcal{T}_{3}\right)$ 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 $\mathcal{C}$.

(a)

(b)

(c)

Fig. 11: (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 $\#_{f}\left(\mathcal{C}_{x}(4)\right)=5$ and $\#_{f}\left(\mathcal{C}_{y}(4)\right)=5$. (c) The Schnyder drawing $s(\mathcal{W})$ of $G$.

Schnyder's algorithm assigns coordinates $(0,0),(2 n-5,0)$, and $(0,2 n-5)$ 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. 11b. Moreover, $\mathcal{P}_{1}(w), \mathcal{P}_{2}(w)$, and $\mathcal{P}_{3}(w)$ have $w$ as the only common vertex [52]. 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 $\left(u_{1}, u_{3}\right)$. 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 $\left(u_{1}, u_{2}\right)$. Then Schnyder's algorithm assigns coordinates $\left(\#_{f}\left(\mathcal{C}_{x}(w)\right), \#_{f}\left(\mathcal{C}_{y}(w)\right)\right.$ ) to $w$; see Fig. 11c.

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 [52].

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 $s\left(\mathcal{W}_{1}\right)$ and $s\left(\mathcal{W}_{2}\right)$ are not the same drawing) and that it is surjective (that is, for any Schnyder drawing $\Gamma$, there exists a Schnyder wood $\mathcal{W}$ such that $s(\mathcal{W})$ is $\Gamma$ ). That $s$ is surjective is actually obvious, as a Schnyder drawing $\Gamma$ 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 $\Gamma$. By definition of Schnyder drawing, there is at least one Schnyder $\operatorname{wood} \mathcal{W}$ such that $s(\mathcal{W})=\Gamma$. We prove that there is at most one such Schnyder wood, that is, $\Gamma$ uniquely determines $\mathcal{W}$.


Fig. 12: Slopes of the edges in a Schnyder drawing on (a) a triangular grid and (b) a square grid.

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 $\mathcal{W}=\left(\mathcal{T}_{1}, \mathcal{T}_{2}, \mathcal{T}_{3}\right)$ 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 $\left(0^{\circ}, 60^{\circ}\right),\left(120^{\circ}, 180^{\circ}\right)$, and $\left(240^{\circ}, 300^{\circ}\right)$, respectively, while the edges of $\mathcal{T}_{1}, \mathcal{T}_{2}$, and $\mathcal{T}_{3}$ outgoing from $v$ have slopes in the intervals $\left(180^{\circ}, 240^{\circ}\right),\left(300^{\circ}, 360^{\circ}\right)$, and $\left(60^{\circ}, 120^{\circ}\right)$, respectively; see, e.g., [27] and Fig. 12a. Hence, back on the square grid, a Schnyder drawing constructed from a Schnyder wood $\mathcal{W}=\left(\mathcal{T}_{1}, \mathcal{T}_{2}, \mathcal{T}_{3}\right)$ 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 $\left(0^{\circ}, 90^{\circ}\right),\left(135^{\circ}, 180^{\circ}\right)$, and $\left(270^{\circ}, 315^{\circ}\right)$, respectively, while the edges of $\mathcal{T}_{1}, \mathcal{T}_{2}$, and $\mathcal{T}_{3}$ outgoing from $v$ have slopes in the intervals $\left(180^{\circ}, 270^{\circ}\right),\left(315^{\circ}, 360^{\circ}\right)$, and $\left(90^{\circ}, 135^{\circ}\right)$, respectively; see Fig. 12b. 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 $\Gamma$. This concludes the proof that $s$ is an injective function and hence the proof of Theorem 8.

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 $4 n-8$ times, since there are $4 n-8$ 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.

## 6 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 this 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.

## References

1. Md. Jawaherul Alam, Therese Biedl, Stefan Felsner, Michael Kaufmann, Stephen G. Kobourov, and Torsten Ueckerdt. Computing cartograms with optimal complexity. Discret. Comput. Geom., 50(3):784-810, 2013.
2. Patrizio Angelini, Steven Chaplick, Sabine Cornelsen, Giordano Da Lozzo, and Vincenzo Roselli. Morphing triangle contact representations of triangulations. Discret. Comput. Geom., 2023. To appear. doi:10.1007/ s00454-022-00475-9.
3. David Avis and Komei Fukuda. Reverse search for enumeration. Discret. Appl. Math., 65(1-3):21-46, 1996. doi: 10.1016/0166-218X (95) 00026-N.
4. Imre Bárány and Günter Rote. Strictly convex drawings of planar graphs. Documenta Math., 11:369-391, 2006.
5. Jérémy Barbay, Luca Castelli Aleardi, Meng He, and J. Ian Munro. Succinct representation of labeled graphs. Algorithmica, 62(1-2):224-257, 2012.
6. Valmir Carneiro Barbosa and Jayme Luiz Szwarcfiter. Generating all the acyclic orientations of an undirected graph. Inf. Process. Lett., 72(1-2):71-74, 1999. doi:10.1016/S0020-0190 (99)00120-9.
7. Ken Been, Eli Daiches, and Chee-Keng Yap. Dynamic map labeling. IEEE Trans. Vis. Comput. Graph., 12(5):773-780, 2006.
8. Garrett Birkhoff. Rings of sets. Duke Mathematical Journal, 3(3):443 - 454, 1937. doi:10.1215/ S0012-7094-37-00334-X.
9. 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.
10. A. Bondy and U.S.R. Murty. Graph Theory. Graduate Texts in Mathematics. Springer London, 2011. URL: https://books.google.it/books?id=HuDFMwZOwcsC.
11. Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, Dominique Poulalhon, and Gilles Schaeffer. Planar graphs, via well-orderly maps and trees. Graphs Comb., 22(2):185-202, 2006.
12. Prosenjit Bose, Vida Dujmovic, Ferran Hurtado, Stefan Langerman, Pat Morin, and David R. Wood. A polynomial bound for untangling geometric planar graphs. Discret. Comput. Geom., 42(4):570-585, 2009.
13. 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.
14. Enno Brehm. 3-orientations and Schnyder 3-tree-decompositions. Master's thesis, Freie Universität Berlin, 2000.
15. Jason Cantarella, Robert B. Kusner, and John M. Sullivan. On the minimum ropelength of knots and links. Inventiones Mathematicae, 150:257-286, 2002.
16. Luca Castelli Aleardi. Algorithms for Graphs on Surfaces: From Graph Drawing to Graph Encoding. Habilitation thesis, Université de Paris, 2021.
17. 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.
18. 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.
19. Alessio Conte, Roberto Grossi, Andrea Marino, and Romeo Rizzi. Efficient enumeration of graph orientations with sources. Discret. Appl. Math., 246:22-37, 2018. doi:10.1016/j.dam.2017.08.002.
20. 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.
21. 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.
22. Hubert de Fraysseix and Patrice Ossona de Mendez. On topological aspects of orientations. Discrete Math., 229(1-3):57-72, 2001.
23. Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. On triangle contact graphs. Comb. Probab. Comput., 3:233-246, 1994.
24. Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. Bipolar orientations revisited. Discret. Appl. Math., 56(2-3):157-179, 1995. doi:10.1016/0166-218X(94)00085-R.
25. 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 '98), pages 426-433. ACM, 1988.
26. Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. Combinatorica, 10(1):41-51, 1990.
27. Raghavan Dhandapani. Greedy drawings of triangulations. Discret. Comput. Geom., 43(2):375-392, 2010.
28. Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall, 1999.
29. Giuseppe Di Battista, Roberto Tamassia, and Luca Vismara. Output-sensitive reporting of disjoint paths. Algorithmica, 23(4):302-340, 1999.
30. Vida Dujmovic, David Eppstein, Matthew Suderman, and David R. Wood. Drawings of planar graphs with few slopes and segments. Comput. Geom., 38(3):194-212, 2007.
31. Stefan Felsner. Convex drawings of planar graphs and the order dimension of 3-polytopes. Order, 18(1):19-37, 2001.
32. Stefan Felsner. Lattice structures from planar graphs. Electron. J. Comb., 11(1), 2004. doi:10.37236/1768.
33. Stefan Felsner and Florian Zickfeld. Schnyder woods and orthogonal surfaces. Discret. Comput. Geom., 40(1):103126, 2008.
34. Robert Ganian, Petr Hlinený, Joachim Kneis, Daniel Meister, Jan Obdrzálek, Peter Rossmanith, and Somnath Sikdar. Are there any good digraph width measures? J. Comb. Theory, Ser. B, 116:250-286, 2016.
35. Michel Habib, Raoul Medina, Lhouari Nourine, and George Steiner. Efficient algorithms on distributive lattices. Discret. Appl. Math., 110(2-3):169-187, 2001. doi:10.1016/S0166-218X (00) 00258-4.
36. Joel Hass and J. C. Lagarias. The number of Reidemeister moves needed for unknotting. J. Amer. Math. Soc., 14:399-428, 2001.
37. Joel Hass, J. C. Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. J. ACM, 46(2):185-211, 1999.
38. Thomas P. Hayes. A simple condition implying rapid mixing of single-site dynamics on spin systems. In 47 th Annual IEEE Symposium on Foundations of Computer Science (FOCS '06), pages 39-46. IEEE Computer Society, 2006.
39. Xin He, Ming-Yang Kao, and Hsueh-I Lu. Linear-time succinct encodings of planar graphs via canonical orderings. SIAM J. Discret. Math., 12(3):317-325, 1999.
40. Arthur B. Kahn. Topological sorting of large networks. Commun. ACM, 5(11):558-562, 1962.
41. Goos Kant. Drawing planar graphs using the canonical ordering. Algorithmica, 16(1):4-32, 1996.
42. Donald E. Knuth. The art of computer programming. Vol. 4A. Combinatorial algorithms. Part 1. Addison-Wesley, 2011.
43. 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.
44. Takao Nishizeki and Md. Saidur Rahman. Planar Graph Drawing, volume 12 of Lecture Notes Series on Computing. World Scientific, 2004.
45. Martin Nöllenburg, Roman Prutkin, and Ignaz Rutter. On self-approaching and increasing-chord drawings of 3-connected planar graphs. J. Comput. Geom., 7(1):47-69, 2016.
46. 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.
47. Gara Pruesse and Frank Ruskey. Gray codes from antimatroids. Order, 10(3):239-252, 1993.
48. Gara Pruesse and Frank Ruskey. Generating linear extensions fast. SIAM J. Comput., 23(2):373-386, 1994.
49. Pierre Rosenstiehl and Robert Endre Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. Discret. Comput. Geom., 1:343-353, 1986.
50. Frank Ruskey. Combinatorial Generation. University of Victoria, 2003.
51. Marcus Schaefer and Daniel Stefankovic. Decidability of string graphs. J. Comput. Syst. Sci., 68(2):319-334, 2004.
52. 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.
53. 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.
54. Matthew B. Squire. Generating the acyclic orientations of a graph. J. Algorithms, 26(2):275-290, 1998. doi: 10.1006/jagm. 1997. 0891.
55. Matthew Blaze Squire. Gray Codes and Efficient Generation of Combinatorial Structures. PhD thesis, North Carolina State University, 1995.
56. 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.
57. Roberto Tamassia, editor. Handbook on Graph Drawing and Visualization. Chapman and Hall/CRC, 2013.
58. Kunihiro Wasa. Enumeration of enumeration algorithms. CoRR, abs/1605.05102, 2016.
59. Hassler Whitney. Non-separable and planar graphs. Trans. Am. Math. Soc., 34(2):339-362, 1932. MR:1501641. Zbl:0004.13103. JFM:58.0608.01. doi:10.2307/1989545.

[^0]:    * Research partially supported by PRIN projects no. 2022ME9Z78 "NextGRAAL: Next-generation algorithms for constrained GRAph visuALization" and no. 2022TS4Y3N "EXPAND: scalable algorithms for EXPloratory Analyses of heterogeneous and dynamic Networked Data".

