Testing Planarity of Partially Embedded Graphs *

Patrizio Angelini†, Giuseppe Di Battista†, Fabrizio Frati†, Vít Jelínek‡§, Jan Kratochvíl‡, Maurizio Patrignani†, and Ignaz Rutter♯

† Dipartimento di Informatica e Automazione, Roma Tre University, Italy
‡ Department of Applied Mathematics, Charles University, Prague, Czech Republic
§ School of Computer Science, Reykjavík University
♯ Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT), Germany

Abstract

We study the following problem: Given a planar graph G and a planar drawing (embedding) of a subgraph of G, can such a drawing be extended to a planar drawing of the entire graph G? This problem fits the paradigm of extending a partial solution to a complete one, which has been studied before in many different settings. Unlike many cases, in which the presence of a partial solution in the input makes hard an otherwise easy problem, we show that the planarity question remains polynomial-time solvable. Our algorithm is based on several combinatorial lemmata which show that the planarity of partially embedded graphs meets the "oncas" behaviour – obvious necessary conditions for planarity are also sufficient. These conditions are expressed in terms of the interplay between (a) rotation schemes and containment relationships between cycles and (b) the decomposition of a graph into its connected, biconnected, and triconnected components. This implies that no dynamic programming is needed for a decision algorithm and that the elements of the decomposition can be processed independently.

Further, by equipping the components of the decomposition with suitable data structures and by carefully splitting the problem into simpler subproblems, we improve our algorithm to reach linear-time complexity.

Finally, we consider several generalizations of the problem, e.g. minimizing the number of edges of the partial embedding that need to be rerouted to extend it, and argue that they are NP-hard. Also, we show how our algorithm can be applied to solve related Graph Drawing problems.

*Work on this problem began at the BICI Workshop on Graph Drawing, held in Bertinoro, Italy, in March 2009, and was carried out while the authors were at the Department of Applied Mathematics, Charles University, Prague.

1 Introduction

Planarity is one of the central concepts not only in Graph Drawing, but in Graph Theory as a whole. The characterization of planar graphs proved by Kuratowski [20] in 1930 marks the beginning of modern Graph Theory. Such a characterization, based on two forbidden topological subgraphs – K_5 and $K_{3,3}$ – makes planarity a finite problem and leads to a polynomial time recognition algorithm. Planarity is thus "simple" from the computational point of view (this, of course, does not mean that algorithms for testing planarity are trivial) in the strongest possible way, as several linear-time algorithms for testing planarity are known [2, 16, 3].

In this paper we pose and study the question of planarity testing in a constrained setting, namely when a part of the input graph is already drawn and cannot be changed. A practical motivation for this question is, e.g., the visualization of large networks in which certain patterns are required to be drawn in a standard way. The known planarity testing algorithms, even those that build a drawing incrementally, are of no help here, since they are allowed to redraw at each step the part of the graph processed so far. For similar reasons, online planar embedding and planarity testing algorithms, such as [28, 25, 4, 24], are not suitable to be used in this context.

The question of testing the planarity of partially drawn graphs fits into the general paradigm of extending a partial solution to a full one. This has been studied in various settings and it often happens that the extendability problem is more difficult than the unconstrained one. As an example, graph coloring is NP-complete for perfect graphs even if only four vertices are already colored [19], while the chromatic number of a perfect graph can be determined in polynomial time. Another example is provided by edge colorings – deciding 3-edge-colorability of cubic bipartite graphs if some edges are already colored is NP-complete [9], while it follows from the famous König-Hall theorem that cubic bipartite graphs are always 3-edge colorable. In view of these hardness results it is somewhat surprising that the planarity of partially drawn graphs can be tested in polynomial time, in fact linear, as we show in this paper. All the more so since this problem is known to be NP-hard [23] for drawings where edges are constrained to be straight-line segments.

Specific constraints on planar graph drawings have been studied by several authors. See, e.g., [27, 26, 6, 14]. Unfortunately, none of those results can be exploited to solve the question we pose in this paper. Mohar [21, 17] gives algorithms for extending 2-cell embeddings on the torus and surfaces of higher genus. However, the 2-cell embedding is a very strong condition that substantially changes the nature of the problem.

In order to solve the general problem, we allow disconnected or low connected graphs to be part of the input. It is readily seen that in this case the rotation schemes (i.e., the cyclic orderings of the edges incident to the vertices of the graph) do not fully describe the input. In fact, the relative position of vertices against cycles in the graph must also be considered. (These concepts and their technical details are discussed later.) Further, we make use of the fact that drawing graphs on the plane and on the sphere are equivalent concepts. The advantage of considering embeddings on the sphere lies in the fact that we do not need to distinguish between the outer face and the inner faces.

The main idea of our algorithm is to look at the problem from the "opposite" perspective. Namely, we do not try to directly extend the input partial embedding (which seems much harder than one would expect). Instead, we look at the possible embeddings of the entire graph and decide if any of them contains the partially embedded part as prescribed by the input.

Our algorithm is based on several combinatorial lemmata, relating the problem to the connectivity of the graph. Most of them exhibit the "oncas" property – the obvious necessary conditions are also sufficient. This is particularly elegant in the case of 2-connected graphs. In this case, we exploit the SPQR-tree decomposition of the graph. This notion was introduced in [4] to describe all the possible embeddings of 2-connected planar graphs in a succinct way and was used in various situations when asking for planar embeddings with special properties. A survey on the use of this technique in planar graphs is [22]. It is indeed obvious that if a 2-connected graph admits a feasible drawing, then the skeleton of each node of the SPQR-tree has a drawing *compatible* (a precise definition of compatibility will come later) with the partial embedding. We prove that the converse is also true. Hence – if we only aim at polynomial running time – we do not need to perform *any* dynamic programming on the SPQR-tree and we could process its nodes independently. However, for the ultimate goal of linear running time, we must refine the approach and pass several information through the SPQR-tree. Then, dynamic programming becomes more than useful. Also, the SPQR-trees are exploited at two levels of abstraction, both for decomposing an entire block and for computing the embedding of the subgraph induced by each face of the constrained part of the drawing.

The paper is organized as follows. In Section 2 we describe the terminology and list auxiliary topological lemmata. In particular, the combinatorial invariants of equivalent embeddings are introduced. In Section 3 we state the combinatorial characterization theorems for disconnected, simply connected, and 2-connected cases. The consequence of them is a simple polynomial-time algorithm outlined at the end of the section. Section 4 is then devoted to describe technical details of the linear-time algorithm. Section 5 summarizes the results, discusses several possible generalizations of the question leading to NP-hard problems, and shows how our techniques can be used to solve other Graph Drawing problems.

2 Notation and Preliminaries

In this section we introduce some notations and preliminaries.

2.1 Drawings, embeddings, and the problem definition A *drawing* of a graph is a mapping of each vertex to a distinct point of the plane and of each edge to a simple Jordan curve connecting its endpoints. A drawing is *planar* if the curves representing its edges do not cross but, possibly, at common endpoints. A graph is *planar* if it admits a planar drawing. A planar drawing Γ determines a subdivision of the plane into connected regions, called *faces*, and a circular ordering of the edges incident to each vertex, called *rotation scheme*. Visiting the (not necessarily connected) border of a face f of Γ in such a way to keep f to the left, we determine a set of circular lists of vertices. Such a set is the *boundary* of f. Two drawings are *equivalent* if they have the same rotation schemes and the same face boundaries. A *planar embedding* is an equivalence class of planar drawings. Let H be a subgraph of a graph G and let $\mathcal{H}$ and $\mathcal{G}$ be embeddings of H and G, respectively. The *restriction* of $\mathcal{G}$ to H is the embedding of H that is obtained from $\mathcal{G}$ by considering only the vertices and the edges of H. Further, $\mathcal{G}$ is an *extension* of $\mathcal{H}$ if the restriction of $\mathcal{G}$ to H is $\mathcal{H}$. We study the following decision problem (see Fig. 1 for an example):

Partially Embedded Planarity (Pep)
Input: A triplet $(G, H, \mathcal{H})$ of graphs G and H, with $H \subseteq G$, and a planar embedding $\mathcal{H}$ of H.

Question: Does G admit a planar embedding whose restriction to H is $\mathcal{H}$?

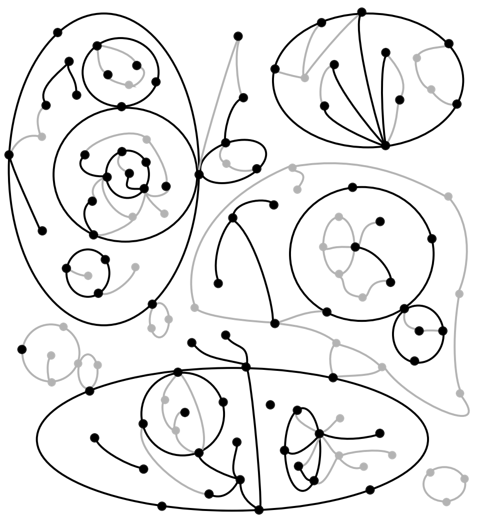

Figure 1: Embedding of a planar graph G whose restriction to H coincides with $\mathcal{H}$. Vertices and edges in H are black; vertices and edges in $G \setminus H$ are grey.

2.2 Connectivity and data structures A graph is *connected* if every pair of vertices is connected by a path. A *k-connected* graph G is such that removing any $k-1$ vertices leaves G connected; 3-connected, 2-connected, and 1-connected graphs are also called *triconnected*, *biconnected*, and *simply connected* graphs, respectively. A *separating k-set* is a set of k vertices whose removal disconnects the graph. Separating 1- and 2-sets are called *cutvertices* and *separation pairs*, respectively. Hence, a connected graph is biconnected if it has no cutvertices and it is triconnected if it has no separation pairs. The maximal biconnected subgraphs of a graph are its *blocks*. Each edge of G falls into a single block of G, while cutvertices are shared by different blocks.

Let $(G, H, \mathcal{H})$ be an instance of Pep. In the following we define some data structures that are widely used throughout the paper. All of such data structures can be easily computed in time linear in the number of edges of the graph or of the embedding they refer to.

The *component-face tree* $\mathcal{CF}$ of $\mathcal{H}$ is a tree whose nodes are the connected components of H and the faces of $\mathcal{H}$. A face f and a component C are joined by an edge if a vertex of C is incident to f. The *block-face tree* $\mathcal{BF}$ of $\mathcal{H}$ is a tree whose nodes are the blocks of H and the faces of $\mathcal{H}$. A face f and a block B are joined by an edge if B contains an edge incident to f. The *vertex-face incidence graph* $\mathcal{VF}$ of $\mathcal{H}$ is a graph whose nodes are the vertices of H and the faces of $\mathcal{H}$. A vertex x and a face f are joined by an edge if x appears on the boundary of f. The *block-cutvertex tree* of a connected graph G is a tree whose nodes are the blocks and the cutvertices of G. Edges in the block-cutvertex tree join each cutvertex to the blocks it belongs to. The *enriched block-cutvertex tree* of a connected graph G is a tree obtained by adding to the block-cutvertex tree of G each vertex v of G that is not a cutvertex and by connecting v to the unique block it belongs to.

The *SPQR-tree* $\mathcal{T}$ of a biconnected graph G describes the arrangement of its triconnected components. In the following, we summarize basic concepts about SPQR-trees. More details can be found in [4].

A graph is *st-biconnectible* if adding edge (s, t) to it yields a biconnected graph. Let G be an st-biconnectible graph. A *split pair* $\{u, v\}$ of G is either a separation pair or a pair of adjacent vertices. A *maximal split component* of G with respect to a split pair $\{u, v\}$ (or, simply, a maximal split component of $\{u, v\}$) is either edge (u, v) or a maximal subgraph G' of G such that G' contains u and v and $\{u, v\}$ is not a split pair of G'. We call *split component* of $\{u, v\}$ the union of any number of maximal split components of $\{u, v\}$.

The SPQR-tree $\mathcal{T}$ of a biconnected graph G rooted at edge e describes a recursive decomposition of G induced by its split pairs. The nodes of $\mathcal{T}$ are of four types: S, P, Q, and R. Their connections are called *arcs*.

Each node μ of $\mathcal{T}$ has an associated st-biconnectible multigraph, called the *skeleton* of μ, denoted by $sk(\mu)$, and showing how the children of μ, represented by "virtual edges", are arranged into μ. The virtual edge in $sk(\mu)$ corresponding to a child node ν, is called the *virtual edge of ν in $sk(\mu)$*.

Recursively replacing each virtual edge e_i of $sk(\mu)$ with the skeleton $sk(\mu_i)$ of its corresponding child μ_i produces a subgraph of G that is called the *pertinent graph* of μ and is denoted by $pert(\mu)$.

Given a biconnected graph G and a reference edge $e = (u', v')$, tree $\mathcal{T}$ is recursively defined as follows. At each step, a split component G^*, a pair of vertices $\{u, v\}$, and a node ν in $\mathcal{T}$ are given. A node μ corresponding to G^* is introduced in $\mathcal{T}$ and attached to its parent ν. Vertices u and v are called the *poles* of μ and also denoted by $u(\mu)$ and $v(\mu)$, respectively. The decomposition possibly recurs on some split components of G^*. At the beginning of the decomposition $G^* = G -$

$\{e\}$, $\{u,v\} = \{u',v'\}$, and ν is a Q-node corresponding to e.

Base Case: If G^* consists of exactly one edge between u and v, then μ is a Q-node whose skeleton is G^* itself.

Parallel Case: If G^* is composed of at least two maximal split components $G_1,\dots,G_k$ ($k \geq 2$) of G with respect to $\{u,v\}$, then μ is a P-node. Graph $sk(\mu)$ consists of k parallel virtual edges between u and v, denoted by $e_1,\dots,e_k$ and corresponding to $G_1,\dots,G_k$, respectively. The decomposition recurs on $G_1,\dots,G_k$, with $\{u,v\}$ as pair of vertices for every graph, and with μ as parent node.

Series Case: If G^* is composed of exactly one maximal split component of G with respect to $\{u,v\}$ and if G^* has cutvertices $c_1,\dots,c_{k-1}$ ($k \geq 2$), appearing in this order on a path from u to v, then μ is an S-node. Graph $sk(\mu)$ is the path $e_1,\dots,e_k$, where virtual edge e_i connects c_{i-1} with c_i ($i = 2,\dots,k-1$), e_1 connects u with c_1, and e_k connects c_{k-1} with v. The decomposition recurs on the split components corresponding to each of $e_1,e_2,\dots,e_{k-1},e_k$ with μ as parent node, and with $\{u,c_1\},\{c_1,c_2\},\dots,\{c_{k-2},c_{k-1}\},\{c_{k-1},v\}$ as pair of vertices, respectively.

Rigid Case: If none of the above cases applies, the purpose of the decomposition step is that of partitioning G^* into the minimum number of split components and recurring on each of them. We need some further definition. Given a maximal split component G' of a split pair $\{s,t\}$ of G^*, a vertex $w \in G'$ *properly belongs* to G' if $w \neq s,t$. Given a split pair $\{s,t\}$ of G^*, a maximal split component G' of $\{s,t\}$ is *internal* if neither u nor v (the poles of G^*) properly belongs to G', *external* otherwise. A *maximal split pair* $\{s,t\}$ of G^* is a split pair of G^* that is not contained into an internal maximal split component of any other split pair $\{s',t'\}$ of G^*. Let $\{u_1,v_1\},\dots,\{u_k,v_k\}$ be the maximal split pairs of G^* ($k \geq 1$) and, for $i = 1,\dots,k$, let G_i be the union of all the internal maximal split components of $\{u_i,v_i\}$. Observe that each vertex of G^* either properly belongs to exactly one G_i or belongs to some maximal split pair $\{u_i,v_i\}$. Node μ is an R-node. Graph $sk(\mu)$ is the graph obtained from G^* by replacing each subgraph G_i with the virtual edge e_i between u_i and v_i. The decomposition recurs on each G_i with μ as parent node and with $\{u_i,v_i\}$ as pair of vertices.

In the following we assume that, for each node μ of the SPQR-tree $\mathcal{T}$ of a graph G, $sk(\mu)$ contains the virtual edge (u,v) representing the parent of μ in $\mathcal{T}$. We say that an edge e of G *projects* to a virtual edge e' of $sk(\mu)$, for some node μ in $\mathcal{T}$, if e belongs to the pertinent graph of the node of $\mathcal{T}$ corresponding to e'.

The SPQR-tree $\mathcal{T}$ of a graph G with n vertices and m edges has $O(n)$ Q-, S-, P-, and R-nodes. Also, the total number of vertices of the skeletons of the nodes of $\mathcal{T}$ is $O(n)$. Graph G is planar if and only if the skeletons of all the nodes of $\mathcal{T}$ are planar [1]. The SPQR-tree $\mathcal{T}$ can be used to represent all the planar embeddings of G. We emphasize the following properties, that are implicitly exploited throughout all the paper.

Property 2.1. *A planar embedding of the skeleton of every node of $\mathcal{T}$ determines a planar embedding of G and vice versa.*

Property 2.2. *Let C be a cycle of G and let μ be any node of $\mathcal{T}$. Then, either the edges of C belong to a single virtual edge of $sk(\mu)$, or they belong to a set of virtual edges that induce a cycle in $sk(\mu)$.*

2.3 Facial cycles and H-bridges

Let Γ be a planar drawing of a graph H (see Fig. 2.a). Let $\vec{C}$ be a simple cycle in H with an arbitrary orientation. The oriented cycle $\vec{C}$ splits the plane into two connected parts. Denote by $V_\Gamma^{left}(\vec{C})$ and $V_\Gamma^{right}(\vec{C})$ the sets of vertices of the graph that are to the left and to the right of $\vec{C}$ in Γ, respectively. The boundary of each face f of Γ can be uniquely decomposed into simple edge-disjoint cycles, bridges (i.e., edges that are not part of a cycle), and isolated vertices (see Fig. 2.b). Orient the cycles in such a way that f is to the left when walking along the cycle according to the orientation. Call these oriented cycles the *facial cycles* of f (see Fig. 2.c). Observe that the sets $V_\Gamma^{left}(\vec{C})$, $V_\Gamma^{right}(\vec{C})$ and the notion of facial cycles only depend on the embedding $\mathcal{H}$ of Γ. Hence, it makes sense to denote $V_\mathcal{H}^{left}(\vec{C})$ and $V_\mathcal{H}^{right}(\vec{C})$, and to consider the facial cycles of $\mathcal{H}$.

Let x be a vertex of a graph G with embedding $\mathcal{G}$. Denote by $E_G(x)$ the set of edges incident to x and by $\sigma_\mathcal{G}(x)$ the rotation scheme of x in $\mathcal{G}$.

Lemma 2.1. *Let $(G,H,\mathcal{H})$ be an instance of* Pep *and let $\mathcal{G}$ be a planar embedding of G. The restriction of $\mathcal{G}$ to H is $\mathcal{H}$ if and only if the following conditions hold: 1) for every vertex $x \in V(H)$, $\sigma_\mathcal{G}(x)$ restricted to $E_H(x)$ coincides with $\sigma_\mathcal{H}(x)$, and 2) for every facial cycle $\vec{C}$ of each face of $\mathcal{H}$, we have that $V_\mathcal{H}^{left}(\vec{C}) \subseteq V_\mathcal{G}^{left}(\vec{C})$ and $V_\mathcal{H}^{right}(\vec{C}) \subseteq V_\mathcal{G}^{right}(\vec{C})$.*

Proof. The proof easily descends from the following statement. Let Γ_1 and Γ_2 be two drawings of the same graph G such that, for every vertex $x \in V(G)$,

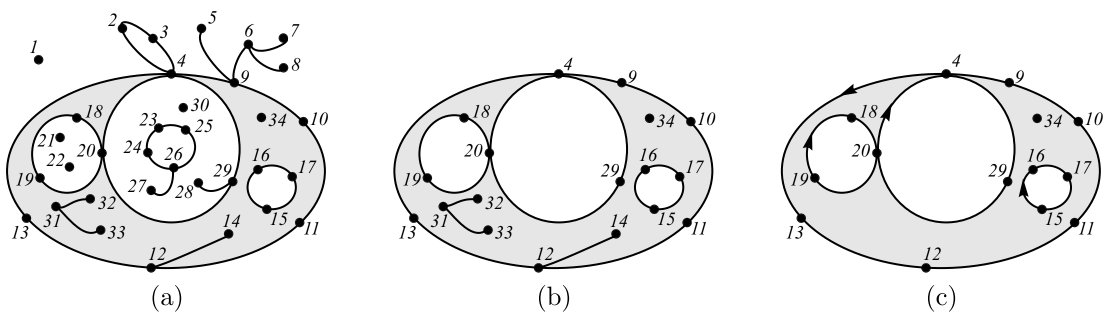

Figure 2: (a) A planar drawing of a graph G. The shaded region represents a face f of the drawing. (b) The boundary of f. The circular lists defining the boundary of f are: $[15, 16, 17]$, $[33, 31, 32, 31]$, $[13, 12, 14, 12, 11, 10, 9, 4, 29, 20, 19, 18, 20, 4]$. (c) The facial cycles of f.

$\sigma_{\Gamma_1}(x) = \sigma_{\Gamma_2}(x)$. Drawings Γ_1 and Γ_2 have the same embedding if and only if Γ_1 and Γ_2 have the same oriented facial cycles and for each facial cycle $\vec{C}$ we have $V^{left}_{\Gamma_1}(\vec{C}) = V^{left}_{\Gamma_2}(\vec{C})$.

We need to prove this statement in both directions: (i) if Γ_1 and Γ_2 have the same embedding then they have the same oriented facial cycles and for each facial cycle we have $V^{left}_{\Gamma_1}(\vec{C}) = V^{left}_{\Gamma_2}(\vec{C})$ and (ii) if Γ_1 and Γ_2 have the same oriented facial cycles and for each facial cycle we have $V^{left}_{\Gamma_1}(\vec{C}) = V^{left}_{\Gamma_2}(\vec{C})$, then Γ_1 and Γ_2 have the same embedding.

The first direction trivially descends from the observation that drawings with the same embedding have the same facial cycles. Suppose for a contradiction that, for some facial cycle $\vec{C}$, $V^{left}_{\Gamma_1}(\vec{C}) \neq V^{left}_{\Gamma_2}(\vec{C})$. Then, at least one vertex v is to the left of $\vec{C}$ in Γ_1 and to the right of $\vec{C}$ in Γ_2 (the opposite case being analogous). Hence, v is part of the boundary of a face that is to the left of $\vec{C}$ in Γ_1 and to the right of $\vec{C}$ in Γ_2, contradicting the hypothesis that Γ_1 and Γ_2 have the same facial boundaries.

For the second direction, first suppose that G is connected and has at least one vertex of degree three. In this case, the fact that Γ_1 and Γ_2 have the same rotation scheme implies that they also have the same face boundaries, and, hence, the same embedding. Second, suppose that G is connected and has maximum degree two. Then, G is either a path or a cycle. In both cases, the face boundaries of Γ_1 and Γ_2 are the same (recall that G is drawn on the sphere). Finally, suppose that G has several connected components C_1, C_2, $\dots$, C_k. Then, Γ_1 and Γ_2 have the same face boundaries if: (a) for each C_i, $i = 1, \dots, k$, the embedding $\mathcal{G}_1$ of Γ_1 restricted to C_i is the same as the embedding $\mathcal{G}_2$ of Γ_2 restricted to C_i and (b) each pair of connected components C_i and C_j, with $i, j = 1, \dots, k$ and $i \neq j$, either do not share a face both in $\mathcal{G}_1$ and in $\mathcal{G}_2$ or they contribute with the same circular lists to the boundary of the same face f in $\mathcal{G}_1$ and in $\mathcal{G}_2$.

Condition (a) is guaranteed as in the two cases in which G is connected. Condition (b) follows from the hypothesis that, for each facial cycle $\vec{C}$, we have $V^{left}_{\Gamma_1}(\vec{C}) = V^{left}_{\Gamma_2}(\vec{C})$. In fact, suppose for a contradiction that two connected components C_x and C_y share a face f in $\mathcal{G}_1$ and no face in $\mathcal{G}_2$. Since C_x and C_y share a face in $\mathcal{G}_1$, they are on the same side of any facial cycle $\vec{C}$ belonging to any other component C_z (more intuitively, C_x and C_y can not be separated by any facial cycle of Γ_1). On the other hand, consider the unique path $C_x, f_1, C_1, f_2, \dots, C_y$ in the component-face tree of $\mathcal{G}_2$. By hypothesis, $C_1 \neq C_x, C_y$. Hence, the facial cycle $\vec{C}$ obtained from the boundary of f_1 and containing vertices of C_1 separates C_x from C_y, thus contradicting the hypothesis that $V^{left}_{\Gamma_1}(\vec{C}) = V^{left}_{\Gamma_2}(\vec{C})$.

Finally, suppose for a contradiction that two connected components C_x and C_y contribute with circular lists L^x_1 and L^y_1 to the boundary of the same face f_1 of $\mathcal{G}_1$ and with circular lists L^x_2 and L^y_2 to the boundary of the same face f_2 of $\mathcal{G}_2$ and suppose that $L^x_1 \neq L^x_2$. The boundary of f_1 is oriented in such a way that every facial cycle has f_1 to its left. Then, every facial cycle obtained from L^x_1 has C_y to its left. Further, for every cycle C' of C_x that is not a facial cycle obtained from L^x_1, there exists a facial cycle $\vec{C}$ obtained from L^x_1 that has C' to its right (part of $\vec{C}$ and of C' may coincide). As $\mathcal{G}_1$ and $\mathcal{G}_2$ restricted to C_x give the same embedding, the last statement is true both in $\mathcal{G}_1$ and in $\mathcal{G}_2$. Then, for every facial cycle $\vec{C'}$ obtained from L^x_2 and not from L^x_1, there exists a facial cycle $\vec{C}$ obtained from L^x_1 hat has $\vec{C'}$ to its right. Since $\vec{C'}$ is incident to f_2 and since C_y is incident to f_2, such a component is to the right of $\vec{C}$, contradicting the hypothesis that $V^{left}_{\Gamma_1}(\vec{C}) = V^{left}_{\Gamma_2}(\vec{C})$. $\square$

Let G be a graph and let H be a subgraph of G. An *H-bridge* K of G is a subgraph of G formed either by a single edge $e \in E(G) \setminus E(H)$ whose end-vertices belong to H or by a connected component K^- of $G - V(H)$, together with all the edges (and their end-vertices) that connect a vertex in K^- to a vertex in H. In the first case, the H-bridge is *trivial*. A vertex that belongs to $V(H) \cap V(K)$ is called an *attachment vertex* (or *attachment*) of K. Note that the edge-sets of the H-bridges form a partition of $E(G) \setminus E(H)$.

An H-bridge K is *local* to a block B of H if all the attachments of K belong to B. Notice that an H-bridge with a single attachment can be local to more than one block, while an H-bridge with at least two attachments is local to at most one block. An H-bridge that is not local to any block of H is *non-local*.

3 Combinatorial Characterization

We first present a combinatorial characterization of the instances of Pep that allow an embedding extension. This not only forms a basis of our algorithm, but it is also interesting in its own right, since it shows that an instance of Pep has an embedding extension if and only if it satisfies simple conditions that are obviously necessary for an embedding extension to exist.

3.1 G biconnected We focus on instances $(G, H, \mathcal{H})$ of Pep in which G is biconnected. This assumption allows us to use the SPQR-tree of G as the main tool of our characterization.

Definition 1. *A planar embedding of the skeleton of a node of the SPQR-tree of G is* edge-compatible with *$\mathcal{H}$ if, for every vertex x of the skeleton and for every three edges of $E_H(x)$ belonging to different virtual edges of the skeleton, their clockwise order determined by the embedding of the skeleton is a suborder of $\sigma_{\mathcal{H}}(x)$.*

Lemma 3.1. *Let $(G, H, \mathcal{H})$ be an instance of* Pep *where G is biconnected. Let $\mathcal{T}$ be the SPQR-tree of G. An embedding $\mathcal{G}$ of G satisfies Condition 1 of Lemma 2.1 if and only if, for each node μ of $\mathcal{T}$, the corresponding embedding of $sk(\mu)$ is edge-compatible with $\mathcal{H}$.*

Proof. Obviously, if G has an embedding satisfying Condition 1 of Lemma 2.1, then the corresponding embedding of $sk(\mu)$ is edge-compatible with $\mathcal{H}$, for each node μ of $\mathcal{T}$.

To prove the converse, assume that the skeleton of every node of $\mathcal{T}$ has an embedding that is edge-compatible with $\mathcal{H}$, and let $\mathcal{G}$ be the embedding of G determined by all such skeleton embeddings. We claim that $\mathcal{G}$ satisfies Condition 1 of Lemma 2.1. To prove the claim, it suffices to show that any three edges e, f, and g of $\mathcal{H}$ that share a common vertex x appear in the same clockwise order around x in $\mathcal{H}$ and in $\mathcal{G}$. Assume that the triple (e, f, g) is embedded in clockwise order around x in $\mathcal{H}$. Let μ be the node of $\mathcal{T}$ with the property that the Q-nodes representing e, f, and g appear in distinct components of $\mathcal{T} - \mu$. Note that such a node μ exists and is unique. The three edges e, f, and g project into three distinct virtual edges e', f', and g' of $sk(\mu)$. Since the embedding of $sk(\mu)$ is assumed to be edge-compatible with $\mathcal{H}$, the triple (e', f', g') is embedded in clockwise order in $sk(\mu)$, and hence the triple (e, f, g) is embedded in clockwise order in $\mathcal{G}$. □

Consider a simple cycle $\vec{C}$ of G with an arbitrary orientation and a node μ of the SPQR-tree of G. Either all the edges of $\vec{C}$ belong to the pertinent graph of a single virtual edge of $sk(\mu)$ or the virtual edges whose pertinent graphs contain the edges of $\vec{C}$ form a simple cycle in $sk(\mu)$. Such a cycle in $sk(\mu)$ inherits the orientation of $\vec{C}$ in a natural way.

Definition 2. *A planar embedding of the skeleton of a node μ of the SPQR-tree of G is* cycle-compatible with *$\mathcal{H}$ if, for every facial cycle $\vec{C}$ of $\mathcal{H}$ whose edges project to a simple cycle $\vec{C}'$ in $sk(\mu)$, all the vertices of $sk(\mu)$ that belong to $V^{left}_{\mathcal{H}}(\vec{C})$ and all the virtual edges that contain vertices of $V^{left}_{\mathcal{H}}(\vec{C})$ (except for the virtual edges of $\vec{C}'$ itself) are embedded to the left of $\vec{C}'$; and analogously for $V^{right}_{\mathcal{H}}(\vec{C})$.*

Lemma 3.2. *Let $(G, H, \mathcal{H})$ be an instance of* Pep *where G is biconnected. Let $\mathcal{T}$ be the SPQR-tree of G. An embedding $\mathcal{G}$ of G satisfies Condition 2 of Lemma 2.1 if and only if, for each node μ of $\mathcal{T}$, the corresponding embedding of $sk(\mu)$ is cycle-compatible with $\mathcal{H}$.*

Proof. Obviously, if $\mathcal{G}$ is an embedding of G that satisfies Condition 2 of Lemma 2.1, then the corresponding embedding of $sk(\mu)$ is cycle-compatible with $\mathcal{H}$, for each node μ of $\mathcal{T}$.

To prove the converse, assume that $sk(\mu)$ has an embedding that is cycle-compatible with $\mathcal{H}$, for each node μ of $\mathcal{T}$, and let $\mathcal{G}$ be the resulting embedding of G.

Our goal is to show that, for every facial cycle $\vec{C}$ of $\mathcal{H}$ and for every vertex x of $H - V(\vec{C})$, the relative left/right position of x with respect to $\vec{C}$ is the same in $\mathcal{H}$ as in $\mathcal{G}$.

Refer to Fig. 3. Assume that x is to the right of $\vec{C}$ in $\mathcal{G}$ (the other case being analogous). Let P be the shortest path in G that connects x to a vertex of $\vec{C}$. Such a path exists since G is connected. Let y be the vertex of $\vec{C} \cap P$, and let e and f be the two edges of $\vec{C}$ adjacent to y, where e directly precedes f in the orientation of $\vec{C}$. By the minimality of P, all the vertices of $P - y$ avoid $\vec{C}$, hence all the vertices of $P - y$ are to the right of $\vec{C}$ in $\mathcal{G}$. Let g be the edge of P adjacent to y. In $\mathcal{G}$, the triple (e, f, g) appears in clockwise order around y.

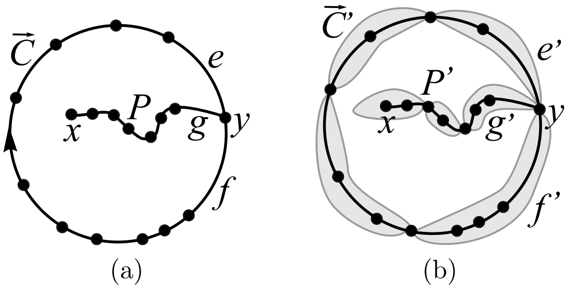

Figure 3: Illustration for the proof of Lemma 3.2. Grey regions represent virtual edges of the skeleton of a node of $\mathcal{T}$.

Let μ be the (unique) internal node of $\mathcal{T}$ in which e, f, and g project to distinct edges e', f', and g' of $sk(\mu)$. Let $\vec{C}'$ be the projection of $\vec{C}$ into $sk(\mu)$ (in other words, $\vec{C}'$ is the subgraph of $sk(\mu)$ formed by edges that contain the projection of at least one edge of $\vec{C}$), and let P' be the projection of P. It is easy to see that $\vec{C}'$ is a cycle of length at least two, while P' is either a path or a cycle. Assume that the edges of $\vec{C}'$ are oriented consistently with the orientation of $\vec{C}$ and that the edges of P' form an ordered sequence, where the edge containing x is the first and g' is the last.

Both the endpoints of an edge of $\vec{C}'$ are vertices of $\vec{C}$. Analogously, both the endpoints of an edge of P' are vertices of P, with the possible exception of the first vertex of P'. It follows that no vertex of P' belongs to $\vec{C}'$, except possibly for the first one and the last one. Thus, no edge of P' belongs to $\vec{C}'$ and, by the assumption that the embedding of $sk(\mu)$ is planar and that $\mathcal{G}$ is the embedding resulting from the skeleton embedding choices, all the edges of P' are embedded to the right of the directed cycle $\vec{C}'$ in $sk(\mu)$. In particular, the edge of $sk(\mu)$ containing x is to the right of $\vec{C}'$. Since the embedding of $sk(\mu)$ is assumed to be cycle-compatible with $\mathcal{H}$, x is to the right of $\vec{C}$ in $\mathcal{H}$.

This shows that $\mathcal{G}$ satisfies Condition 2 of Lemma 2.1, as claimed. □

DEFINITION 3. *A planar embedding of the skeleton of a node μ of the SPQR-tree of G is* compatible with $\mathcal{H}$ *if it is both edge- and cycle-compatible with $\mathcal{H}$.*

As a consequence of Lemmata 3.1 and 3.2, we obtain the following result, characterizing the positive instances of PEP in which G is biconnected.

THEOREM 3.1. *Let $(G, H, \mathcal{H})$ be an instance of* PEP *where G is biconnected. Then G has an embedding which extends $\mathcal{H}$ if and only if the skeleton of each node of its SPQR-tree has an embedding compatible with $\mathcal{H}$.*

If G is biconnected we can use Theorem 3.1 for devising a polynomial time algorithm for PEP. Namely, we can test, for each node μ of the SPQR-tree $\mathcal{T}$ of G whether μ has an embedding compatible with $\mathcal{H}$. For Q-, S-, and R-nodes, this test is easily done in polynomial time.

If μ is a P-node, the test is more complex. Let x and y be the two poles of $sk(\mu)$. We say that a virtual edge e of $sk(\mu)$ is *constrained* if the pertinent graph of e (that is, the pertinent graph of the child node of μ in $\mathcal{T}$ corresponding to e) contains at least one edge of H incident to x and at least one edge of H incident to y. To obtain an embedding of μ edge-compatible with $\mathcal{H}$, the constrained edges must be embedded in a cyclic order that is consistent with $\sigma_{\mathcal{H}}(x)$ and $\sigma_{\mathcal{H}}(y)$. Such a cyclic order, if it exists, is unique and can be determined in polynomial time. Note that, if $\mathcal{H}$ has a facial cycle $\vec{C}$ that projects to a proper cycle $\vec{C}'$ in μ, then $\vec{C}'$ has exactly two edges and these two edges are both constrained. Thus, the embedding of any such cycle $\vec{C}'$ in μ is fixed as soon as we fix the cyclic order of the constrained edges. Once the cyclic order of the constrained edges of μ is determined, we process the remaining edges one-by-one and insert them among the edges that are already embedded, in such a way that no edge-compatibility or cycle-compatibility constraints are violated. It is not difficult to verify that this procedure constructs an embedding of μ compatible with $\mathcal{H}$, if such an embedding exists.

3.2 G simply-connected or disconnected

A graph is planar if and only if each of its blocks is planar. Thus, planarity testing of general graphs can be reduced to planarity testing of biconnected graphs. For partially embedded planarity, the same simple reduction does not work (see Fig. 4). However, we will show that solving partially embedded planarity for a general instance $(G, H, \mathcal{H})$ can be reduced to solving the subinstances induced by the blocks of G and to checking additional conditions that guarantee that the partial solutions can be combined into a full solution for G.

Let us consider instances $(G, H, \mathcal{H})$ of PEP in which G is connected. When dealing with such an instance, it is often useful to assume that G has no non-trivial non-local H-bridge. We will now show that any instance of PEP can be transformed to an equivalent instance that satisfies this additional assumption.

Let K be a non-trivial non-local H-bridge of G. Since K is non-local, it must have at least two attachments, and these attachments do not belong to any single block of H.

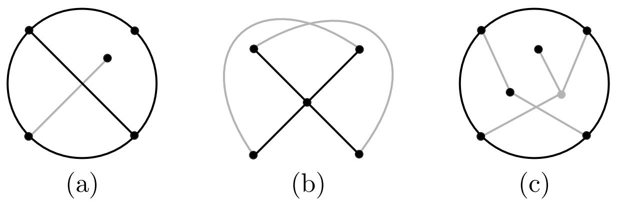
Figure 4: Three examples of PEP instances $(G, H, \mathcal{H})$ that have no embedding extension, even though each block of G admits an embedding extending the corresponding sub-embedding of $\mathcal{H}$. The black edges and vertices represent $\mathcal{H}$, the gray edges and vertices belong to G but not to H. Note that instance (a) fails to satisfy Condition 3 of Lemma 3.3, instance (b) fails to satisfy Condition 2 of Lemma 3.3, and instance (c) has a non-trivial non-local H-bridge.

Let f_K be the face of $\mathcal{H}$ whose boundary contains all the attachments of the H-bridge K. Note that there can be at most one such a face (see Fig. 5.a): If the attachments of K were contained in the intersection of the boundaries of two distinct faces of $\mathcal{H}$, then K would necessarily be local. If there is no face of $\mathcal{H}$ incident to all the attachments of K, then G clearly has no embedding extension (see Fig. 5.b). In this case, we define f_K as an arbitrary face of $\mathcal{H}$.

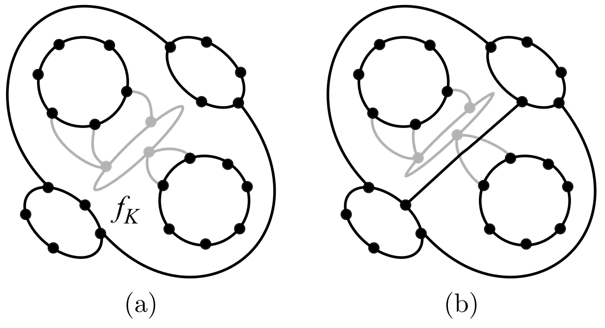

Figure 5: A non-local bridge is either necessarily contained in a face f_K (a) or causes a non-planarity (b).

Let $\mathcal{K}$ be the set of non-trivial non-local H-bridges of G. It is clear that, in any embedding of G extending $\mathcal{H}$, all the vertices of $K - V(H)$ are embedded inside f_K, for every $K \in \mathcal{K}$. This motivates the following definition.

DEFINITION 4. *Let H' be the graph whose edge set is equal to the edge set of H, and whose vertex set is defined by $V(H') = V(H) \cup \bigcup_{K \in \mathcal{K}} V(K)$. Let $\mathcal{H}'$ be the embedding of H' that is obtained from $\mathcal{H}$ by inserting, for every H-bridge $K \in \mathcal{K}$, all the vertices of $K - V(H)$ into the interior of the face f_K.*

Observe that the graph G has no non-trivial non-local H'-bridges. Observe also, that any embedding of G that extends $\mathcal{H}$ also extends $\mathcal{H}'$, and vice versa. Thus, the instance $(G, H, \mathcal{H})$ of PEP is equivalent to the instance $(G, H', \mathcal{H}')$, which contains no non-trivial non-local bridges.

Let $\mathcal{H}$ be an embedding of a graph H, and let H_1 and H_2 be edge-disjoint subgraphs of H. We say that H_1 and H_2 *alternate* around a vertex x of $\mathcal{H}$ if there are two pairs of edges $e, e' \in E(H_1)$ and $f, f' \in E(H_2)$ that are incident to x and that appear in the cyclic order (e, f, e', f') in the rotation scheme of x restricted to these four edges. Let x and y be two vertices of H and let $\vec{C}$ be a directed cycle in $\mathcal{H}$. We say that $\vec{C}$ *separates* x and y if $x \in V_{\mathcal{H}}^{left}(\vec{C})$ and $y \in V_{\mathcal{H}}^{right}(\vec{C})$, or vice versa.

LEMMA 3.3. *Let $(G, H, \mathcal{H})$ be an instance of* PEP *where G is connected and every non-trivial H-bridge of G is local. Let $G_1, \ldots, G_t$ be the blocks of G, let H_i be the subgraph of H induced by the vertices of G_i, and let $\mathcal{H}_i$ be $\mathcal{H}$ restricted to H_i. Then, G has an embedding extending $\mathcal{H}$ if and only if 1) each G_i has an embedding extending $\mathcal{H}_i$, 2) no two distinct graphs H_i and H_j alternate around any vertex of $\mathcal{H}$, and 3) for every facial cycle $\vec{C}$ of $\mathcal{H}$ and for any two vertices x and y of $\mathcal{H}$ separated by $\vec{C}$, any path in G connecting x and y contains a vertex of $\vec{C}$.*

Proof. Clearly, the three conditions of the lemma are necessary. To show that they are also sufficient, assume that the three conditions are satisfied and proceed by induction on the number t of blocks of G.

If $t = 1$, then G is biconnected, and there is nothing to prove. Assume that $t \geq 2$. If there is at least one block G_i that does not contain any vertex of H, we restrict our attention to the subgraph G' of G induced by those blocks that contain at least one vertex of H. Since every non-trivial H-bridge of G is local, graph G' is connected, and hence it satisfies the three conditions of the lemma. By induction, the embedding $\mathcal{H}$ can be extended into an embedding $\mathcal{G}'$ of G'. Since every block G_i of G is planar (by condition 1 of the lemma), it is easy to extend the embedding $\mathcal{G}'$ into an embedding of G.

Assume now that every block of G contains at least one vertex of H. This implies that every cutvertex of G belongs to H, because otherwise the cutvertex would belong to a non-local H-bridge, which is impossible by assumption. Let x be any cutvertex of G. Let $G'_1, G'_2, \ldots, G'_k$ be the connected components of $G - x$, where we select G'_1 by the following rules: if there is a component of $G - x$ that has no vertex connected to x by an edge of H, then let G'_1 be such a component; if each component of $G - x$ is connected to x by an edge of H, then choose G'_1 in such a way that the edges of H incident to x and belonging to G'_1 form an interval in $\sigma_{\mathcal{H}}(x)$. Such a choice of G'_1 is always possible, due to condition 2 of the lemma.

Let G' be the subgraph of G induced by $V(G'_1) \cup \{x\}$, and let G'' be the subgraph of G induced by $V(G'_2) \cup \cdots \cup V(G'_k) \cup \{x\}$. Let H' and H'' be the subgraphs of H induced by the vertices of G' and G'', respectively, and let $\mathcal{H}'$ and $\mathcal{H}''$ be $\mathcal{H}$ restricted to H' and H'', respectively. Both G' and G'' have fewer blocks than G. Also, both the instances $(G', H', \mathcal{H}')$ and $(G'', H'', \mathcal{H}'')$ satisfy the conditions of the lemma. Thus, by induction, there is an embedding $\mathcal{G}'$ of G' that extends $\mathcal{H}'$ and an embedding $\mathcal{G}''$ of G'' that extends $\mathcal{H}''$.

Our goal is to combine $\mathcal{G}'$ and $\mathcal{G}''$ into a single embedding of G that extends $\mathcal{H}$. To see that this is possible, we prove two auxiliary claims.

Claim 1. $\mathcal{H}'$ has a face f' whose boundary contains x and, for any facial cycle $\vec{C}$ of f', all the vertices of H'' except for x are in $V^{left}_{\mathcal{H}}(\vec{C})$, i.e., they are 'inside' f'. To see that the claim holds, assume first that H' has no edge incident to x (see Fig. 6.a). Let f' be the unique face of $\mathcal{H}'$ incident to x. We show that all the vertices of H'' are inside f' in $\mathcal{H}$. Let y be any vertex of H''. Since G'' is connected, there is a path P in G'' from y to x. Assume for contradiction that $\mathcal{H}'$ has a facial cycle $\vec{C}$ such that $\vec{C}$ separates y from x in $\mathcal{H}$. This cycle belongs to $H' - x$, hence $\vec{C}$ and P are disjoint, contradicting condition 3 of the lemma.

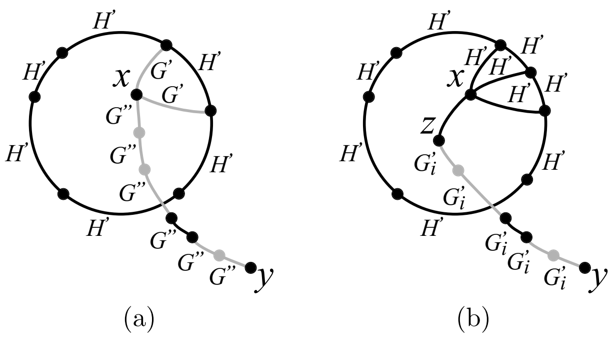

Figure 6: Illustration for the proof of Lemma 3.3. (a) H' has no edge incident to x. (b) H' has an edge incident to x.

Next, assume that H' has an edge incident to x (see Fig. 6.b). By the construction of G_1, each connected component of $G - x$ has at least one vertex connected to x by an edge of H. Moreover, the edges of $\mathcal{H}'$ incident to x form an interval in $\sigma_{\mathcal{H}}(x)$. This shows that $\mathcal{H}'$ has a face f' containing x on its boundary, and such that every vertex of H'' adjacent to x is inside f' in $\mathcal{H}$. We now show that all the vertices of H'' except for x are inside f'. Let y be a vertex of H'' different from x. Let G'_i be the component of $G - x$ containing y. We know that G'_i has a vertex z adjacent to x by an edge of H and that z is inside f' in $\mathcal{H}$. Let P be a path in G'_i connecting y and z. If y is not inside f', then y is separated from z in $\mathcal{H}$ by a facial cycle of $\mathcal{H}'$, contradicting condition 3 of the lemma.

Claim 2. All the vertices of H', except for x, appear in $\mathcal{H}$ inside the same face f'' of $\mathcal{H}''$; further, x is on the boundary of f''. To prove the claim, note that any two vertices from $H' - x$ are inside the same face f'' of $\mathcal{H}''$ in $\mathcal{H}$ by condition 3 of the lemma, because they are connected by a path in G'_1. Vertex x is on the boundary of f'', since otherwise it would be separated in $\mathcal{H}$ from the remaining vertices of H' by a facial cycle of f'', again contradicting condition 3 of the lemma.

In view of the previous two claims, it is easy to see that the embedding $\mathcal{G}'$ of G' and the embedding $\mathcal{G}''$ of G'' can be combined into a single embedding $\mathcal{G}$ of G that extends $\mathcal{H}$. To see this, note that, when $\mathcal{H}'$ is extended into $\mathcal{G}'$, the face f' from Claim 1 can be subdivided into several faces of $\mathcal{G}'$, at least one of which, say g', contains x on its boundary. Analogously, the face f'' from Claim 2 can be subdivided into several faces of $\mathcal{G}''$, at least one of which, say g'', contains x on its boundary. We then obtain the embedding $\mathcal{G}$ by merging the faces g' and g'' into a single face. $\square$

Observe that the second and third conditions of Lemma 3.3 can be easily checked in polynomial time.

Next, we focus on the instances $(G, H, \mathcal{H})$ of Pep in which G is not connected. The possibility of solving the subinstances of $(G, H, \mathcal{H})$ induced by the connected components of G does not guarantee that instance $(G, H, \mathcal{H})$ of Pep has a solution. However, we show that solving Pep for an instance $(G, H, \mathcal{H})$ can be reduced to solving the subinstances induced by the connected components of G and to checking additional conditions that guarantee that the partial solutions can be combined into a full solution for G.

Lemma 3.4. *Let $(G, H, \mathcal{H})$ be an instance of* Pep. *Let $G_1, \dots, G_t$ be the connected components of G. Let H_i be the subgraph of H induced by the vertices of G_i, and let $\mathcal{H}_i$ be $\mathcal{H}$ restricted to H_i. Then G has an embedding extending $\mathcal{H}$ if and only if: 1) each G_i has an embedding extending $\mathcal{H}_i$, and 2) for each i, for each facial cycle $\vec{C}$ of H_i and for every $j \neq i$, no two vertices of H_j are separated by $\vec{C}$.*

Proof. Clearly, the two conditions of the lemma are necessary. To show that they are also sufficient, assume that the two conditions are satisfied and proceed by induction on the number t of connected components of G.

If $t = 1$ then G is connected and there is nothing to prove. Assume now that G has $t \geq 2$ connected components $G_1, \ldots, G_t$. Let H_i and $\mathcal{H}_i$ be defined as in the statement of the lemma. Let $\mathcal{CF}$ be the component-face tree of $\mathcal{H}$, rooted at a node that represents an arbitrary face of $\mathcal{H}$. We say that a face f_i of $\mathcal{H}$ is the *outer face* of $\mathcal{H}_i$ if at least one child of f_i in $\mathcal{CF}$ is a component of H_i, but the parent of f_i is not a component of H_i. Observe that, due to the second condition of the lemma, each $\mathcal{H}_i$ has exactly one outer face f_i. We thus have a sequence of (not necessarily distinct) outer faces $f_1, \ldots, f_t$.

Let us now assume, without loss of generality, that in the subtree of $\mathcal{CF}$ rooted at f_1, there is no outer face $f_i \neq f_1$. This implies that f_1 is the only face of $\mathcal{H}$ that is incident both to H_1 and to $H - H_1$. By induction, the embedding $\mathcal{H} - \mathcal{H}_1$ can be extended to an embedding $\mathcal{G}_{\geq 2}$ of the graph $G - G_1$. By the first condition of the lemma, $\mathcal{H}_1$ can be extended into an embedding $\mathcal{G}_1$ of G_1. The two embeddings $\mathcal{H} - \mathcal{H}_1$ and $\mathcal{H}_1$ share a single face f_1.

When extending the embedding $\mathcal{H}_1$ into $\mathcal{G}_1$, the face f_1 of $\mathcal{H}_1$ can be subdivided into several faces of $\mathcal{G}_1$. Let f' be any face of $\mathcal{G}_1$ obtained by subdividing f_1. Analogously, in the embedding $\mathcal{G}_{\geq 2}$ the face f_1 can be subdivided into several faces, among which we choose an arbitrary face f''.

We then glue the two embeddings $\mathcal{G}_1$ and $\mathcal{G}_{\geq 2}$ by identifying face f' of $\mathcal{G}_1$ and face f'' of $\mathcal{G}_{\geq 2}$ into a single face whose boundary is the union of the boundaries of f' and f''. This yields an embedding of G that extends $\mathcal{H}$. □

Note that the second condition of Lemma 3.4 can be easily tested in polynomial time. Thus, we can use the characterization to directly prove that PEP is solvable in polynomial time. In the rest of the paper, we describe a more sophisticated algorithm that solves PEP in linear time.

4 Linear-Time Algorithm

In this section we show a linear time algorithm for solving PEP. First, we tackle the case in which G is biconnected. The algorithm solving this case, presented in Subsection 4.3, uses the algorithms presented in Subsections 4.1 and 4.2 as subroutines. Then, we deal with the case in which G is simply connected and with the general case in Subsection 4.4. Some data structures are exploited by the algorithm we present, namely block-cutvertex trees, SPQR-trees, enriched block-cutvertex trees, block-face trees, component-face trees, and vertex-face incidence graphs. These data structures can be easily computed in linear time [15].

4.1 G biconnected, H biconnected In this section we show how to solve PEP in linear time if both G and H are biconnected.

LEMMA 4.1. *Let $(G, H, \mathcal{H})$ be an instance of* PEP *such that both G and H are biconnected. Let $\mathcal{G}$ be any planar embedding of G satisfying Condition 1 of Lemma 2.1. Then, $\mathcal{G}$ satisfies Condition 2 of Lemma 2.1.*

Proof. Suppose, for a contradiction, that a planar embedding $\mathcal{G}$ of G exists such that $\mathcal{G}$ satisfies Condition 1 and does not satisfy Condition 2 of Lemma 2.1. Then, there exists a facial cycle $\vec{C}$ of $\mathcal{H}$ such that either there exists a vertex $x \in V_{\mathcal{H}}^{left}(\vec{C})$ with $x \in V_{\mathcal{G}}^{right}(\vec{C})$ or there exists a vertex $x \in V_{\mathcal{H}}^{right}(\vec{C})$ with $x \in V_{\mathcal{G}}^{left}(\vec{C})$. Suppose that we are in the former case, as the latter case can be discussed analogously. Since $\mathcal{H}$ is a planar embedding and H is connected, there exists a path $P = (x_1, x_2, \ldots, x_k) \in H$ such that x_1 is a vertex of $\vec{C}$, $x_i \in V_{\mathcal{H}}^{left}(\vec{C})$, for each $i = 2, \ldots, k$, and $x_k = x$. Denote by x_1^- and by x_1^+ the vertex preceding and following x_1 in the oriented cycle $\vec{C}$, respectively. Consider the placement of x_2 with respect to $\vec{C}$ in $\mathcal{G}$. As $x_2 \notin \vec{C}$, either $x_2 \in V_{\mathcal{G}}^{left}(\vec{C})$ or $x_2 \in V_{\mathcal{G}}^{right}(\vec{C})$. In the first case, path $(x_2, \ldots, x_k)$ crosses $\vec{C}$, since $x_2 \in V_{\mathcal{G}}^{left}(\vec{C})$, $x_k \in V_{\mathcal{G}}^{right}(\vec{C})$, and no vertex v_i belongs to $\vec{C}$, for $i = 2, \ldots, k$, thus contradicting the planarity of the embedding $\mathcal{G}$. In the second case, the clockwise order of the edges incident to x_1 in $\mathcal{H}$ is (x_1, x_1^-), (x_1, x_2), and (x_1, x_1^+), while the clockwise order of the edges incident to x_1 in $\mathcal{G}$ is (x_1, x_1^-), (x_1, x_1^+), and (x_1, x_2), thus contradicting the assumption that $\mathcal{G}$ satisfies Condition 1 of Lemma 2.1. □

Due to Lemma 4.1, testing whether a planar embedding $\mathcal{G}$ exists satisfying Conditions 1 and 2 of Lemma 2.1 is equivalent to testing whether a planar embedding $\mathcal{G}$ exists satisfying Condition 1 of Lemma 2.1. Due to Lemma 3.1, testing whether a planar embedding $\mathcal{G}$ exists satisfying Condition 1 is equivalent to testing whether the skeleton of each node of the SPQR-tree of G has a planar embedding that is edge-compatible with $\mathcal{H}$.

Algorithm BB Construct the SPQR-tree $\mathcal{T}$ of G and root it at an arbitrary internal node. A bottom-up visit of $\mathcal{T}$ is performed. After a node μ of $\mathcal{T}$ has been visited, an embedding of $sk(\mu)$ that is edge-compatible with $\mathcal{H}$ is selected, if it exists.

In order to keep track of the edges of H that belong to $pert(\mu)$ and that are incident to the poles $u(\mu)$ and

$v(\mu)$, define the *first edge* $f_{u(\mu)}$ and the *last edge* $l_{u(\mu)}$ (the *first edge* $f_{v(\mu)}$ and the *last edge* $l_{v(\mu)}$) as the edges of H such that all and only the edges between $f_{u(\mu)}$ and $l_{u(\mu)}$ (resp. between $f_{v(\mu)}$ and $l_{v(\mu)}$) in the counterclockwise order of the edges incident to $u(\mu)$ (resp. to $v(\mu)$) in $\mathcal{H}$ belong to $pert(\mu)$. After a node μ of $\mathcal{T}$ has been visited by the algorithm, edges $f_{u(\mu)}$, $l_{u(\mu)}$, $f_{v(\mu)}$, and $l_{v(\mu)}$ are associated with μ. Refer also to $f_{u(e)}$ and $l_{u(e)}$ (resp. $f_{v(e)}$ and $l_{v(e)}$) where e is the virtual edge corresponding to μ in the skeleton of the parent of μ.

If μ is a Q- or an S-node, no check is needed. As $sk(\mu)$ is a cycle, the only planar embedding of $sk(\mu)$ is edge-compatible with $\mathcal{H}$. Edges $f_{u(\mu)}$, $l_{u(\mu)}$, $f_{v(\mu)}$, and $l_{v(\mu)}$ are easily computed.

If μ is an R-node, then $sk(\mu)$ has only two planar embeddings. For each of them, verify if it is edge-compatible with $\mathcal{H}$ by performing the following check. For each vertex x of $sk(\mu)$ restrict the circular list of its incident virtual edges to the virtual edges $e_1, \dots, e_h$ that contain an edge of H incident to x. Check if $l_{x(e_i)}$ precedes $f_{x(e_{i+1})}$ (for $i = 1, \dots, h$, where $e_{h+1} = e_1$) in the list of the edges incident to x in $\mathcal{H}$. If x is a pole, do an analogous check on the linear list of its incident virtual edges obtained by removing the virtual edge corresponding to the parent of μ from the circular list. If one of the tests succeeds, then select the corresponding embedding for $sk(\mu)$. Set $f_{u(\mu)} = f_{u(f_1)}$, $l_{u(\mu)} = l_{u(f_p)}$, $f_{v(\mu)} = f_{v(g_1)}$, and $l_{v(\mu)} = l_{v(g_q)}$, where f_1 and f_p (g_1 and g_q) are the first and the last virtual edge in the linear list of the virtual edges containing an edge of H and incident to $u(\mu)$ (resp. to $v(\mu)$).

If μ is a P-node, an embedding of $sk(\mu)$ is a counterclockwise order of its virtual edges around $u(\mu)$. We describe how to verify if an embedding of $sk(\mu)$ exists edge-compatible with $\mathcal{H}$. Consider the virtual edges containing edges of H incident to $u(\mu)$. Construct a list L_u of such edges corresponding to the ordering they have in any embedding of $sk(\mu)$ edge-compatible with $\mathcal{H}$. Insert any of such edges, say e_i, into L_u. Repeatedly consider the last element e_j of L_u and insert as last element of L_u edge e_{j+1} such that $l(u(e_j))$ immediately precedes $f(u(e_{j+1}))$ in the counterclockwise order of the edges incident to $u(\mu)$ in $\mathcal{H}$. If $e_{j+1} = e_i$, then L_u is the desired circular list. If e_{j+1} does not exist, then the edge following $l(u(e_j))$ belongs to the virtual edge corresponding to the parent of μ. Then, consider again edge e_i. Repeatedly consider the first element e_j of L_u and insert as first element of L_u edge e_{j-1} such that $f(u(e_j))$ immediately follows $l(u(e_{j-1}))$ in the counterclockwise order of the edges incident to $u(\mu)$ in $\mathcal{H}$. If e_{j-1} does not exist, then check whether all the virtual edges containing edges of H incident to $u(\mu)$ have been processed and in such a case insert the virtual edge corresponding to the parent of μ as first element of L_u. Analogously construct a list L_v. Let L_{uv} be the sublist obtained by restricting L_u to those edges that appear in L_v. Let L_{vu} be the corresponding sublist of L_v. Check whether L_{uv} and L_{vu} are the reverse of each other. If this is the case, a list L of the virtual edges of $sk(\mu)$ containing edges of H incident to $u(\mu)$ or to $v(\mu)$ can be easily constructed compatible with both L_u and L_v. Finally, arbitrarily insert into L the virtual edges of $sk(\mu)$ not in L_u and not in L_v, thus obtaining an embedding of $sk(\mu)$ edge-compatible with $\mathcal{H}$. Denote by f_1 and f_p (by g_1 and g_q) the virtual edges containing edges of H incident to $u(\mu)$ (resp. to $v(\mu)$) following and preceding the virtual edge representing the parent of μ in L. Set $f_{u(\mu)} = f_{u(f_1)}$, $l_{u(\mu)} = l_{u(f_p)}$, $f_{v(\mu)} = f_{v(g_1)}$, and $l_{v(\mu)} = l_{v(g_q)}$.

THEOREM 4.1. *Let $(G, H, \mathcal{H})$ be an n-vertex instance of* PEP *such that both G and H are biconnected. Algorithm BB solves* PEP *for $(G, H, \mathcal{H})$ in $O(n)$ time.*

Proof. We show that Algorithm BB processes each node μ of $\mathcal{T}$ in $O(k_\mu)$ time, where k_μ is the number of children of μ in $\mathcal{T}$.

First, observe that the computation of $f_{u(\mu)}$, $l_{u(\mu)}$, $f_{v(\mu)}$, and $l_{v(\mu)}$ is trivially done in $O(1)$ time once the embedding of $sk(\mu)$ has been decided.

If μ is a Q-node or an S-node, Algorithm BB does not perform any check or embedding choice.

If μ is an R-node, Algorithm BB computes the two planar embeddings of $sk(\mu)$ in $O(k_\mu)$ time. For each of such embeddings, Algorithm BB processes each node x of $sk(\mu)$ separately, considering the list of the virtual edges incident to x (which is trivially constructed in $O(t)$ time, where t is the number of such edges), and restricting the list to those virtual edges containing an edge of H incident to x (for each virtual edge, it suffices to check whether the first edge incident to x is associated with an edge of H, which is done in $O(1)$ time). Checking whether $l_{x(e_i)}$ precedes $f_{x(e_{i+1})}$ in the list of the edges incident to x in $\mathcal{H}$ is done in $O(1)$ time. Hence, the total time spent for each node x is $O(t)$. Summing up over all the nodes of $sk(\mu)$ results in a total $O(k_\mu)$ time, as every edge is incident to two nodes and the total number of edges in $sk(\mu)$ is $O(k_\mu)$.

If μ is a P-node, extracting the virtual edges of $sk(\mu)$ containing edges of H incident to $u(\mu)$ or to $v(\mu)$ can be done in $O(k_\mu)$ time, as in the R-node case. For each of such edges, equipping $f_{u(e)}$, $l_{u(e)}$, $f_{v(e)}$, and $l_{v(e)}$ with a link to e is done in constant time. Determining an ordering of the virtual edges containing edges of H incident to $u(\mu)$ can be done in $O(k_\mu)$ time, as the operations performed for each virtual edge e_i are accessing to the first and the last edge of e_i, accessing to the edge following the last edge of e_i (preceding the first edge of e_i) in the counterclockwise order of the edges incident to $u(\mu)$ in $\mathcal{H}$, accessing to a virtual edge linked from a first or last edge; each of such operations is trivially done in $O(1)$ time. Marking the virtual edges in L_u and in L_v is done in $O(k_\mu)$ time, as L_u and L_v have $O(k_\mu)$ elements. Then, obtaining L_{uv} and L_{vu}, and checking whether they are the reverse of each other is done in $O(k_\mu)$ time. Finally, extending L_{uv} to L is also easily done in $O(k_\mu)$ time; namely, if L_{uv} is empty, then let L be the concatenation of L_u and L_v (where such lists are made linear by cutting them at any point). Otherwise, start from an edge e_i of L_{uv}; e_i is also in L_u and in L_v; insert e_i into L; insert into L all the edges of L_u following e_i till the next edge e_{i+1} of L_{uv} has been found; insert into L all the edges of L_v preceding e_i till the next edge e_{i+1} of L_{uv} has been found; insert e_{i+1} into L, and repeat the procedure. Each element of L_{uv}, L_u, and L_v is visited once, hence such a step is performed in $O(k_\mu)$ time.

As $\sum_{\mu \in \mathcal{T}} k_\mu = O(n)$, the total running time of the algorithm is $O(n)$. □

4.2 G biconnected, all the vertices and edges of G are in the same face f of $\mathcal{H}$

The instances of Pep considered in this section are denoted by $(G(f), H(f), \mathcal{H}(f))$. Such instances are assumed to satisfy the following properties: (i) $G(f)$ is biconnected, (ii) $G(f)$ and $H(f)$ have the same vertex set, (iii) all the vertices and edges of $H(f)$ are incident to the same face f of $\mathcal{H}(f)$, and (iv) no edge of $G(f) \setminus H(f)$ connects two vertices of the same block of $H(f)$. Algorithm BF, that deals with such a setting, is used as a subroutine by Algorithm BA, to be shown later, dealing with the instances of Pep in which G is biconnected and H arbitrary.

Algorithm BF As in Algorithm BB, the SPQR-tree $\mathcal{T}(f)$ of $G(f)$ is constructed and rooted at an arbitrary internal node. Tree $\mathcal{T}(f)$ is visited bottom-up. After a node μ of $\mathcal{T}$ has been visited, an embedding of $sk(\mu)$ that is compatible with $\mathcal{H}(f)$ is selected, if it exists.

Edges $f_{u(\mu)}$, $l_{u(\mu)}$, $f_{v(\mu)}$, and $l_{v(\mu)}$ of a node μ of $\mathcal{T}(f)$ (and of a virtual edge e) are defined as in Algorithm BB. After each node μ of $\mathcal{T}(f)$ is visited, a flag $p(\mu)$ is set equal to TRUE if there exists a *traversing path* P, that is, a path between $u(\mu)$ and $v(\mu)$ that is composed of edges of $H(f)$, that belongs to $pert(\mu)$, and that is part of a simple cycle $\vec{C}$ of $H(f)$ not entirely contained in $pert(\mu)$; flag $p(\mu)$ is set equal to FALSE otherwise. If $p(\mu) =$ TRUE, a flag $uv(\mu)$ is set equal to TRUE if P is oriented from $u(\mu)$ to $v(\mu)$ according to the orientation of $\vec{C}$, and it is set equal to FALSE otherwise. Refer also to $p(e)$ and $uv(e)$, where e is the virtual edge corresponding to μ in the skeleton of the parent of μ.

We state some useful lemmata. The first one exploits the particular structure of the input to simplify the test of cycle-compatibility with $\mathcal{H}(f)$ for the skeleton of a node μ of $\mathcal{T}(f)$.

Lemma 4.2. *Consider any node μ of $\mathcal{T}(f)$. Then, an embedding of $sk(\mu)$ is cycle-compatible with $\mathcal{H}(f)$ if and only if, for every facial cycle $\vec{C}$ of $\mathcal{H}(f)$ whose edges project to a cycle $\vec{C}'$ of $sk(\mu)$, no vertex and no edge of $sk(\mu)$ is to the right of $\vec{C}'$, where $\vec{C}'$ is oriented according to the orientation of $\vec{C}$.*

Proof. By assumption (iii) of the input, all the vertices and edges of $H(f)$ are incident to the same face f of $\mathcal{H}(f)$. By construction, every facial cycle $\vec{C}$ of $H(f)$ is oriented in such a way that f and hence all the vertices of $H(f)$ are to the left of $\vec{C}$. Then, by Lemma 3.2, if the edges of $\vec{C}$ determine a cycle $\vec{C}'$ of virtual edges of $sk(\mu)$, all the vertices of $sk(\mu)$ that are not in $\vec{C}$ and all the virtual edges of $sk(\mu)$ that are not in $\vec{C}'$ and that contain vertices of $G(f)$ have to be to the left of $\vec{C}'$. Finally, all the virtual edges that are not in $\vec{C}'$ and that do not contain any vertex of $G(f)$ (that is, virtual edges corresponding to Q-nodes) have one end-vertex that is not in $\vec{C}$, by assumption (iv) of the input. Such an end-vertex forces the edge to be to the left of $\vec{C}'$. □

The next property relates the edges of $H(f)$ to the cycles of such a graph.

Property 4.1. *Every simple path of $H(f)$ belongs to at most one simple cycle of $H(f)$.*

Proof. Suppose that there exists a path (that can possibly be a single edge) of $H(f)$ belonging to at least two simple cycles of $H(f)$. Then, such cycles define at least three regions of the plane. Not all the edges of the two cycles can be incident to the same region, contradicting the fact that all the edges of $H(f)$ are incident to the same region of the plane in $\mathcal{H}(f)$. □

Now, we state lemmata specifically dealing with S-, R-, and P-nodes of $\mathcal{T}(f)$.

Lemma 4.3. *Let μ be an S-node of $\mathcal{T}(f)$ with children $\mu_1, \mu_2, \ldots, \mu_k$. Then, $p(\mu_i) =$ TRUE for some $1 \le i \le k$ if and only if $p(\mu_j) =$ TRUE for all $1 \le j \le k$.*

Proof. If $p(\mu_j) =$ TRUE for all $1 \le j \le k$, then trivially $p(\mu_i) =$ TRUE. If $p(\mu_i) =$ TRUE for some $1 \le i \le k$, there exists a traversing path of μ_i that is part of a simple cycle $\vec{C}$ of $H(f)$ not entirely contained in $pert(\mu_i)$;

however, as μ is an S-node, $\vec{C}$ does not entirely lie inside $pert(\mu)$, as otherwise it would entirely lie inside $pert(\mu_i)$. Then, $\vec{C}$ consists of a traversing path of $pert(\mu_j)$, for all $1 \leq j \leq k$, and of a traversing path of the virtual edge of $sk(\mu)$ corresponding to the parent of μ in $\mathcal{T}(f)$, thus proving the lemma. □

LEMMA 4.4. *Let μ be an R-node of $\mathcal{T}(f)$. If an edge e of $sk(\mu)$ has a traversing path belonging to a facial cycle $\vec{C}$, let us orient e in the direction determined by the projection of $\vec{C}$ in $sk(\mu)$. An embedding of $sk(\mu)$ is cycle-compatible with $\mathcal{H}(f)$ if and only if, for each face g of the embedding of $sk(\mu)$, either (i) every virtual edge e on the boundary of g is oriented so that g is to the right of e, or (ii) none of the virtual edges on the boundary of g is oriented in a way that g is to the right of it.*

Proof. Suppose that an embedding of $sk(\mu)$ is cycle-compatible with $\mathcal{H}(f)$. Let g be a face of the embedding. Assume that on the boundary of g there is an edge e containing a traversing path P, such that g is to the right of e. Let $\vec{C}$ be the facial cycle of $\mathcal{H}(f)$ that contains P. By Lemma 4.2, $\vec{C}$ projects to a directed cycle $\vec{C}'$ in $sk(\mu)$, and no vertex or edge of $sk(\mu)$ is embedded to the right of $\vec{C}'$. Thus, $\vec{C}'$ corresponds to the boundary of the face g, and hence g satisfies condition (i).

Suppose now that in an embedding of $sk(\mu)$, every face satisfies condition (i) or condition (ii). We claim that the embedding of $sk(\mu)$ is cycle-compatible with $\mathcal{H}(f)$. To prove it, we use Lemma 4.2. Let $\vec{C}$ be a facial cycle of $\mathcal{H}(f)$ that projects to a simple cycle $\vec{C}'$ in $sk(\mu)$. Let e be any edge of $\vec{C}'$ and let g be the face to the right of e in the embedding of $sk(\mu)$. Necessarily, g satisfies condition (i). Hence, each edge on the boundary of g has a traversing path. The union of these paths forms a cycle in $\mathcal{H}(f)$, and by Property 4.1, this cycle is equal to $\vec{C}$. Thus, the boundary of g coincides with the cycle $\vec{C}'$. In particular, no vertex and no edge of $sk(\mu)$ is embedded to the right of $\vec{C}'$. By Lemma 4.2, this means that the embedding of $sk(\mu)$ is cycle-compatible with $\mathcal{H}(f)$. □

LEMMA 4.5. *Let μ be a P-node of $\mathcal{T}(f)$. There exist either zero or two virtual edges of $sk(\mu)$ containing a traversing path.*

Proof. If there exists one virtual edge e_i of $sk(\mu)$ containing a traversing path that is part of a simple cycle $\vec{C}$ of $H(f)$ not entirely contained in $pert(e_i)$, another virtual edge of $sk(\mu)$ containing a traversing path that is part of $\vec{C}$ exists, as otherwise $\vec{C}$ would not be a cycle. Further, if there exist at least three virtual edges of $sk(\mu)$ containing traversing paths, then each of such paths belongs to three simple cycles, thus contradicting Property 4.1. □

We are now ready to exhibit the main steps of Algorithm BF.

If μ is a Q- or an S-node, no check is needed. As $sk(\mu)$ is a cycle, the only planar embedding of $sk(\mu)$ is compatible with $\mathcal{H}(f)$. Edges $f_{u(\mu)}$, $l_{u(\mu)}$, $f_{v(\mu)}$, and $l_{v(\mu)}$, and flags $p(\mu)$ and $uv(\mu)$ can be easily computed.

If μ is an R-node, for each of the two planar embeddings of $sk(\mu)$, check if it is edge-compatible with $\mathcal{H}(f)$ and set values for $f_{u(\mu)}$, $l_{u(\mu)}$, $f_{v(\mu)}$, and $l_{v(\mu)}$ as in Algorithm BB. In order to check if any of the two embeddings is cycle-compatible with $\mathcal{H}(f)$, check if Lemma 4.2 is satisfied. To perform such a test, first determine if the virtual edge e_p of $sk(\mu)$ representing the parent of μ in $\mathcal{T}(f)$ contains a traversing path P_p and, in case it does, determine its orientation. By definition of traversing path, P_p exists if and only if there exists a traversing path in $pert(\mu)$. Restrict $sk(\mu)$ to those edges $e_i \neq e_p$ with $p(e_i) =$ TRUE and denote by $sk'(\mu)$ the obtained graph. Check if the degree of $u(\mu)$ and $v(\mu)$ in $sk'(\mu)$ is even or odd. In the former case, P_p does not exist; set $p(\mu) =$ FALSE and $p(e_p) =$ FALSE. In the latter case, P_p exists; set $p(\mu) =$ TRUE and $p(e_p) =$ TRUE; the orientation of P_p is the only one that makes the number of edges e_i incident to $u(\mu)$ with $uv(e_i) =$ TRUE equal to the number of edges e_i incident to $u(\mu)$ with $uv(e_i) =$ FALSE; this determines $uv(\mu)$ and $uv(e_p)$. Now, $p(e_i)$ and $uv(e_i)$ are defined for every virtual edge e_i of $sk(\mu)$. Consider every face g of $sk(\mu)$ and denote by $e_j = (u_j, v_j)$ any edge incident to g. Suppose, without loss of generality, that g is to the right of e_j when traversing such an edge from u_j to v_j. Then, check if $p(e_j) =$ FALSE, or $p(e_j) =$ TRUE and $uv(e_j) =$ FALSE, for all edges e_j incident to g, and check whether $p(e_j) =$ TRUE and $uv(e_j) =$ TRUE, for all edges e_j incident to g. If one of the two checks succeeds, the face does not violate Lemma 4.2, otherwise it does.

If μ is a P-node, check if an embedding of $sk(\mu)$ exists that is compatible with $\mathcal{H}(f)$ as follows. By Lemma 4.5, there exist either zero or two virtual edges of $sk(\mu)$ containing a traversing path. Then, consider the children μ_i of μ such that $p(\mu_i) =$ TRUE. If zero or two such children exist, then the edge of $sk(\mu)$ corresponding to the parent ν of μ in $\mathcal{T}$ has no traversing path; if one such a child exists, then the edge of $sk(\mu)$ corresponding to ν has a traversing path. Denote by e_i and e_j the edges of $sk(\mu)$ containing a traversing path, if such edges exist, where possibly e_j corresponds to ν (in this case, set $p(e_j) =$ TRUE, and set $uv(e_j) =$ TRUE if $uv(e_i) =$ FALSE and $uv(e_j) =$ FALSE otherwise). If there exists no edge e_i of $sk(\mu)$ such that $p(e_i) =$ TRUE, then construct an embedding of $sk(\mu)$ that is edge-compatible with $\mathcal{H}(f)$, if possible, as in Algorithm BB; as there exists no facial cycle of $\mathcal{H}(f)$ whose edges belong to distinct edges of $sk(\mu)$, then an edge-compatible embedding is also cycle-compatible with $\mathcal{H}(f)$. Edges $f_{u(\mu)}$, $l_{u(\mu)}$, $f_{v(\mu)}$, and $l_{v(\mu)}$ are computed as in Algorithm BB. Flag $p(\mu) =$ FALSE. If there exist two edges e_i and e_j such that $p(e_i) =$ TRUE, $p(e_j) =$ TRUE, and $p(e_l) =$ FALSE for every edge $e_l \neq e_i, e_j$, suppose that $uv(e_i) =$ TRUE and $uv(e_j) =$ FALSE, the case in which $uv(e_i) =$ FALSE and $uv(e_j) =$ TRUE being analogous. Then, by Lemma 4.2, e_j has to immediately precede e_i in the counterclockwise order of the edges incident to $u(\mu)$. Then, construct L_u and L_v as in Algorithm BB; check whether L_u and L_v, restricted to the edges that appear in both lists, are the reverse of each other; further, check whether e_j precedes e_i in L_u and whether e_i precedes e_j in L_v; if the checks are positive construct the list L of all the edges of $sk(\mu)$ as in Algorithm BB, except for the fact that the edges of $sk(\mu)$ not in L_u and not in L_v are not inserted between e_j and e_i. Edges $f_{u(\mu)}$, $l_{u(\mu)}$, $f_{v(\mu)}$, and $l_{v(\mu)}$ are computed as in Algorithm BB. Set $p(\mu) =$ FALSE if e_j corresponds to a child μ_j of μ and $p(\mu) =$ TRUE if e_j corresponds to the parent of μ in $\mathcal{T}$; in the latter case, $uv(\mu) =$ TRUE if $uv(\mu_i) =$ TRUE and $uv(\mu) =$ FALSE otherwise. We get the following:

THEOREM 4.2. *Let $(G(f), H(f), \mathcal{H}(f))$ be an n-vertex instance of* PEP *such that $G(f)$ is biconnected, $G(f)$ and $H(f)$ have the same vertex set, all the vertices and edges of $H(f)$ are incident to the same face f of $\mathcal{H}(f)$, and no edge of $G(f) \setminus H(f)$ connects two vertices belonging to the same block of $H(f)$. Algorithm BF solves* PEP *for $(G(f), H(f), \mathcal{H}(f))$ in $O(n)$ time.*

Proof. We show that Algorithm BF processes each node μ of $\mathcal{T}(f)$ in $O(k_\mu)$ time, where $\mu_1, \dots, \mu_{k_\mu}$ are the children of μ in $\mathcal{T}(f)$.

Observe that the computation of $f_{u(\mu)}$, $l_{u(\mu)}$, $f_{v(\mu)}$, and $l_{v(\mu)}$ and the check of edge-compatibility are done as in Algorithm BB, hence they take $O(k_\mu)$ time. We describe how to check the cycle-compatibility of an embedding of $sk(\mu)$ in $O(k_\mu)$ time.

If μ is a Q-node or an S-node, Algorithm BF does not perform any check nor embedding choice.

If μ is a P-node, then Algorithm BF performs the same checks and embedding choices as Algorithm BB, plus the check that the two edges e_i and e_j with $p(e_i) =$ TRUE and $p(e_j) =$ TRUE (notice that one of such edges could be the virtual edge corresponding to the parent of μ) are consecutive (with the right order) in L_u and L_v, that is done in constant time. Flags $p(\mu)$ and $uv(\mu)$ are computed in $O(k_\mu)$ time, by simply checking the flags $p(\mu_i)$ and $uv(\mu_i)$, for $i = 1, \dots, k$.

Suppose that μ is an R-node. The construction of $sk'(\mu)$ can be easily done in $O(k_\mu)$ time, as such a graph can be obtained from $sk(\mu)$ by simply checking flag $p(e_i)$, for each edge e_i in $sk(\mu)$. Then, the degree of $u(\mu)$ and $v(\mu)$ in $sk'(\mu)$, and the flags $p(\mu)$, $uv(\mu)$, $p(e_p)$ and $uv(e_p)$ can be computed in total $O(k_\mu)$ time. The test on each face takes time linear in the number of edges incident to the face. Namely, such a test consists of two checks, each of which requires to consider a constant number of flags associated with each edge of the face. As every edge is incident to two faces of $sk(\mu)$ and the number of edges in $sk(\mu)$ is $O(k_\mu)$, the total time spent for the test on the faces of $sk(\mu)$ is $O(k_\mu)$.

As $\sum_{\mu \in \mathcal{T}} k_\mu = O(n)$, the total running time of the algorithm is $O(n)$. □

4.3 G biconnected We show how to solve PEP in the case in which G is biconnected and H is arbitrary. The algorithm we propose is as follows. First, compute a subgraph H^+ of G with the following properties: (i) H^+ is biconnected; (ii) H is a subgraph of H^+; (iii) H^+ contains every non-local H-bridge of G. Second, solve instance $(H^+, H, \mathcal{H})$ obtaining an embedding $\mathcal{H}^+$ of H^+ extending $\mathcal{H}$, if H^+ admits one. Finally, solve instance $(G, H^+, \mathcal{H}^+)$ with Algorithm BB.

Let H' and $\mathcal{H}'$ be as in Definition 4. Let f be a face of $\mathcal{H}'$ and let $V(f)$ be the set of vertices of H' that are incident to f. Let $H(f)$ be the subgraph of H' induced by $V(f)$ and let $\mathcal{H}(f)$ be $\mathcal{H}'$ restricted to $H(f)$. Let H^+ be the graph obtained from G by removing the vertices and edges (but not the attachments) of all the local H-bridges of G. Notice that H^+ has the same vertex set as H'. Let $G(f)$ be the subgraph of H^+ induced by $V(f)$. Note that any embedding of H^+ that extends $\mathcal{H}$ also extends $\mathcal{H}'$ and vice versa. Furthermore, in any embedding of H^+ that extends $\mathcal{H}$, the edges of $G(f)$ not belonging to $H(f)$ are embedded inside f.

LEMMA 4.6. *H^+ is biconnected.*

Proof. By construction of H^+, each H^+-bridge of G has all its attachment vertices in the same block of H, and hence in the same block of H^+, as H is a subgraph of H^+. Therefore, the number of blocks of H^+ is not modified by the addition of the H^+-bridges of G. Since such an addition produces G, which is biconnected, H^+ is biconnected. □

LEMMA 4.7. *An instance $(G, H, \mathcal{H})$ of* PEP *such that G is biconnected admits a solution if and only if (a) instance $(H^+, H, \mathcal{H})$ admits a solution and (b) for every such a solution $\mathcal{H}^+$, instance $(G, H^+, \mathcal{H}^+)$ admits a solution.*

Proof. Clearly, if conditions *(a)* and *(b)* hold, then G has an embedding extending $\mathcal{H}$.

To prove the converse, assume that G has an embedding $\mathcal{G}$ extending $\mathcal{H}$. Clearly, $\mathcal{G}$ contains a sub-embedding $\mathcal{H}^+$ of H^+ that extends $\mathcal{H}$, so condition *(a)* holds. It remains to prove that condition *(b)* holds, too.

First, we introduce some terminology: Let f be any face of $\mathcal{H}$ and let $\mathcal{H}^+$ be any embedding of H^+ that extends $\mathcal{H}$. In $\mathcal{H}^+$, the face f can be partitioned into several faces, which we will call the *subfaces of* f. A set of vertices $S \subseteq V(H)$ is said to be *mutually visible in* f *with respect to* $\mathcal{H}^+$ if $\mathcal{H}^+$ has a subface of f that contains all the vertices of S on its boundary.

The proof that condition *(b)* holds is based on two claims. The first one shows that for the vertices that belong to the same block of H, mutual visibility is independent of the choice of $\mathcal{H}^+$:

Claim 1. Let $\vec{C}$ *be a facial cycle of* f *and let* $S \subseteq V(\vec{C})$ *be a set of vertices of* $\vec{C}$*. If the vertices in* S *are mutually visible in* f *with respect to at least one embedding of* H^+ *that extends* $\mathcal{H}$*, then they are mutually visible in* f *with respect to every embedding of* H^+ *that extends* $\mathcal{H}$*.*

Note that the mutual visibility of S in f only depends on the embedding $\mathcal{H}^+$ restricted to $G(f)$. Let $\mathcal{T}$ be the SPQR-tree of $G(f)$. By Theorem 3.1, the embeddings of $G(f)$ that extend $\mathcal{H}(f)$ are exactly obtained by specifying a compatible embedding for each skeleton of $\mathcal{T}$. Assume that $\mathcal{G}_1$ and $\mathcal{G}_2$ are two embeddings of $G(f)$ that extend $\mathcal{H}$. Assume that the vertices of S are mutually visible in f with respect to $\mathcal{G}_1$. We will show that they are also mutually visible with respect to $\mathcal{G}_2$. In view of Theorem 3.1, we may assume that $\mathcal{G}_2$ was obtained from $\mathcal{G}_1$ by changing the embedding of the skeleton of a single node $\mu \in \mathcal{T}$.

Let us distinguish two cases, depending on whether the cycle $\vec{C}$ is contained in the pertinent graph of a single edge of μ, or whether it projects to a cycle in μ. If $\vec{C}$ is part of the pertinent graph of a single virtual edge $e = \{x, y\} \in \mu$, then let $\mathcal{G}_e$ be the embedded graph obtained as the union of the pertinent graph of e and a single edge connecting x and y, embedded in the outer face of the pertinent graph. We easily see that the vertices S are mutually visible in f if and only if they share the same face of $\mathcal{G}_e$, other than the face that is to the right of $\vec{C}$. Since $\mathcal{G}_e$ does not depend on the embedding of μ, we see that S are mutually visible in $\mathcal{G}_2$.

Assume now that the cycle $\vec{C}$ projects to a cycle $\vec{C}'$ in μ. By Lemma 4.2, in any compatible embedding of μ, all the vertices and edges of μ that do not belong to $\vec{C}'$ are embedded to the left of $\vec{C}'$. In particular, if μ is an R-node, it only has a single compatible embedding. Thus, μ must be a P-node. Let e and e' be the two virtual edges of μ that form $\vec{C}'$. In each compatible embedding of μ, these two edges must be embedded next to each other, and in the same order. It easily follows that any two compatible embeddings of μ yield embeddings of $G(f)$ in which the vertices from S have the same mutual visibility. This completes the proof of the claim.

Let us proceed with the proof that condition *(b)* holds. We need more terminology: Let K and K' be a pair of local H-bridges of G whose attachments all appear on a facial cycle $\vec{C}$ of a face f in $\mathcal{H}$. We say that K and K' have a *three-vertex conflict on* $\vec{C}$ if they share at least three attachments, and that they have a *four-vertex conflict on* $\vec{C}$ if there are four vertices x, x', y, y' which appear on $\vec{C}$ in this cyclic order, and x, y are attachments of K, while x', y' are attachments of K'.

Claim 2. Assume that a face f_K *of* $\mathcal{H}$ *has been assigned to every local* H*-bridge* K *of* G *so that all the attachments of* K *are on the boundary of* f_K*. Let* $\mathcal{H}^+$ *be an embedding of* H^+ *extending* $\mathcal{H}$*. There is an embedding* $\mathcal{G}$ *of* G *extending* $\mathcal{H}^+$*, with the additional property that each local* H*-bridge* K *is embedded inside a subface of* f_K*, if and only if:*

1. *For any local* H*-bridge* K*, all the attachments of* K *are mutually visible in* f_K *with respect to* $\mathcal{H}^+$*.*
2. *If* K *and* L *are distinct local* H*-bridges assigned to the same face* $f_K = f_L$*, such that the attachments of* K *and* L *appear on a common facial cycle* $\vec{C}$ *of* $\mathcal{H}^+$*, then* K *and* L *have no conflict on* $\vec{C}$*.*

Clearly, the two conditions are necessary. In order to prove that they are also sufficient, assume that both the conditions hold. Construct an embedding of $\mathcal{G}$ with the desired properties as follows. Let f be any face of $\mathcal{H}$ and let f' be a face of $\mathcal{H}^+$ which is a subface of f. Let $K_1, \dots K_s$ be all the local H-bridges that were assigned to f and such that all their attachments appear on the boundary of f'. Observe that the first condition of the claim guarantees that every H-bridge K_i can be assigned to a face f' such that all the attachments of K_i are mutually visible in f'. We show that all the bridges $K_1, \dots K_s$ can be embedded inside f'.

First, observe that the boundary of f' is a simple cycle C', because H^+ is biconnected. Observe also that no two bridges K_i and K_j have a conflict on C', by the second condition of the claim. To show that all the bridges $K_1, \dots, K_s$ can be embedded inside C', proceed by induction on s. If $s = 1$ the statement is clear. Assume that $s \geq 2$ and that the bridge K_1 has been successfully embedded into f'. The embedding of K_1 partitions f' into several subfaces $f'_1, \dots, f'_t$. Such subfaces are again bounded by simple cycles, otherwise

G would not be biconnected. We claim that, for every bridge K_i, with $i \geq 2$, there is a subface f'_j containing all the attachments of K_i. Consider any bridge K_i. Assume first that K_i has an attachment x that is not an attachment of K_1. Then, x belongs to a unique subface f'_j. Hence, if K_i has another attachment not belonging to f'_j, there is a four-vertex conflict of K_1 and K_i on $\vec{C}'$, contradicting the second condition of the claim. Assume next that each attachment of K_i is also an attachment of K_1. Then, K_i has exactly two attachments and, if such attachments do not share a face f'_j, a four-vertex conflict of K_1 and K_i on $\vec{C}'$ is created, again contradicting the second condition of the claim.

We can thus assign to each K_i a subface f'_j that contains all its attachments. By induction, all the K_i's can be embedded into their assigned faces, thus proving the second claim.

The proof that condition *(b)* holds easily follows from the two claims. Namely, assume that G has an embedding $\mathcal{G}$ extending $\mathcal{H}$. Let $\mathcal{H}^+$ be $\mathcal{G}$ restricted to H^+. For every local H-bridge K of G, let f_K be the face of $\mathcal{H}$ inside which K is embedded in $\mathcal{G}$. Clearly, $\mathcal{H}^+$ satisfies the two conditions of the second claim, since it can be extended into $\mathcal{G}$. Then, every embedding of H^+ that extends $\mathcal{H}$ satisfies the two conditions of the second claim: For the first condition, this is a consequence of the first claim and for the second condition this is obvious. We conclude that every embedding of H^+ that extends $\mathcal{H}$ can be extended into an embedding of G, thus proving condition *(b)* and hence the lemma. □

Algorithm BA Starting from an instance $(G, H, \mathcal{H})$ of Pep, graphs $G(f)$ and $H(f)$, and embedding $\mathcal{H}(f)$, for every face f of $\mathcal{H}$, are computed as follows. For each H-bridge K of G, determine whether it is local to a block of H or not. In the former case, K is not associated to any face f of $\mathcal{H}$. In the latter case, we compute the unique face f of $\mathcal{H}$ in which K has to be embedded in any solution of instance $(G, H, \mathcal{H})$ of Pep and we associate K with f. Such computations involve checks on the $\mathcal{CF}$-tree of $\mathcal{H}$, on the $\mathcal{BF}$-tree of $\mathcal{H}$, on the $\mathcal{VF}$-graph of $\mathcal{H}$, and on the enriched block-cutvertex tree of each connected component of H. However, all such a computation can be performed in time linear in the size of K, as shown in the following.

Lemma 4.8. *Let $(G, H, \mathcal{H})$ be any instance of* Pep. *Let K be an H-bridge of G. There is an algorithm that checks whether K is local to any block of H in time linear in the size of K. Further, if K is non-local, such an algorithm computes the only face of $\mathcal{H}$ incident to all the attachment vertices of K, if such a face exists, in time linear in the size of K.*

Proof. Compute the component-face tree $\mathcal{CF}$ of $\mathcal{H}$, rooted at any node, the vertex-face incidence graph $\mathcal{VF}$ of $\mathcal{H}$, the block-face tree $\mathcal{BF}$ of $\mathcal{H}$, rooted at any node, and, for each connected component C_i of H, the enriched block-cut vertex tree $\mathcal{B}_i^+$ of C_i, rooted at any node. Such computations can be performed in linear time (as shows in the Data Structures section).

Consider the attachment vertices $a_1, a_2, \dots, a_h$ of K. If $h = 1$, then K is local. Otherwise, $h \geq 2$. In order to decide whether K is local for some block of H, we perform the following check. Consider the attachment vertices a_1 and a_2. If a_1 and a_2 belong to distinct connected components, then K is not local to any block. Otherwise, they belong to the same connected component C_i. Check whether a_1 and a_2 have distance 2 in $\mathcal{B}_i^+$, that is, whether they belong to the same block B. This can be done in constant time [18]. If the check fails, then K is not local to any block. Otherwise, B contains both a_1 and a_2. In the latter case, check whether B is also adjacent in $\mathcal{B}_i^+$ to all the other attachment vertices $a_3, \dots, a_h$ of K. Again, each such a check is performed in constant time [18]. If the test succeeds, then K is local to block B. Otherwise, there exists a vertex a_j, with $3 \leq j \leq h$, that is not incident to B, and K is not local to any block.

If K is non-local, we compute the unique face f of $\mathcal{H}$ to which all the attachment vertices of K are incident. Consider two attachment vertices a_p and a_q, with $1 \leq p, q \leq h$, that do not belong to the same block. Observe that, if a_1 and a_2 do not belong to the same block, then $a_p = a_1$ and $a_q = a_2$. If the check failed on an attachment vertex a_j in $a_3, \dots, a_h$, then either a_1 and a_j, or a_2 and a_j do not belong to the same block. In the former case set $a_p = a_1$ and $a_q = a_j$, in the latter one $a_p = a_2$ and $a_q = a_j$. Since the vertex-face incidence graph $\mathcal{VF}$ is planar, we may use the approach of [18] to determine in constant time whether a_p and a_q are connected by a path of length two in $\mathcal{VF}$, and find the middle vertex of such a path. This middle vertex corresponds to the unique common face f of a_p and a_q. Check whether all the attachments of K are adjacent to f in $\mathcal{VF}$. If the test fails, then no face of $\mathcal{H}$ contains all the attachments of K. Otherwise, f is the only face of $\mathcal{H}$ whose boundary contains all the attachments of K. □

For each face f of $\mathcal{H}$, consider every H-bridge K associated with f. Add the vertices and the edges of K to $G(f)$, and add the vertices of K to $\mathcal{H}(f)$ inside f. Let $H^+ = \bigcup_{f \in H} G(f)$. For each face f of $\mathcal{H}$ call Algorithm BF with input $(G(f), H(f), \mathcal{H}(f))$. If Algorithm BF succeeds for every instance $(G(f), H(f), \mathcal{H}(f))$ (thus providing an embedding $\mathcal{H}^+(f)$ of $G(f)$ whose restriction to $H(f)$ is $\mathcal{H}(f)$), merge the embeddings $\mathcal{H}^+(f)$ of

$G(f)$ into a planar embedding $\mathcal{H}^+$ of H^+. Finally, call Algorithm BB with $(G, H^+, \mathcal{H}^+)$.

THEOREM 4.3. *Let $(G, H, \mathcal{H})$ be an n-vertex instance of* PEP *such that G is biconnected. Algorithm BA solves* PEP *for $(G, H, \mathcal{H})$ in $O(n)$ time.*

Proof. The correctness of the algorithm descends from Lemma 4.7.

By Lemma 4.8, determining whether an H-bridge K is local or not can be done in time linear in the size of K. Further, if K is non-local, the only face of $\mathcal{H}$ incident to all the attachment vertices of K can be computed, if it exists, in time linear in the size of K. Then, the construction of graphs $G(f)$, $H(f)$, H^+ and of embeddings $\mathcal{H}(f)$ takes $O(n)$ time, as it only requires to perform the union of graphs that have total $O(n)$ edges.

By Theorem 4.2, Algorithm BF runs in time linear in the number of edges of $G(f)$, hence all the executions of Algorithm BF take a total $O(n)$ time. By Theorem 4.1, Algorithm BB runs in $O(n)$ time, hence the total running time of Algorithm BA is $O(n)$. □

4.4 G simply connected or disconnected First, we deal with instances $(G, H, \mathcal{H})$ of PEP in which G is simply connected, every non-trivial H-bridge of G is local, and H is arbitrary. We show that the three conditions of Lemma 3.3 can be checked in linear time. The first condition can be checked in linear time by Lemma 4.8. The second and the third conditions can be checked in linear time by the following two lemmata.

LEMMA 4.9. *Let $(G, H, \mathcal{H})$ be an instance of* PEP, *where G is connected. Let $G_1, \dots, G_t$ be the blocks of G, and let H_i be the subgraph of H induced by the vertices of G_i. There is a linear-time algorithm that checks whether any two distinct graphs among $H_1, \dots, H_t$ alternate around a vertex of $\mathcal{H}$.*

Proof. Let us describe the algorithm that performs the required checks. We assume that every edge e of H has an associated label indicating the block of G that contains e. We also associate to each block two integer counters which will be used in the algorithm.

We now describe a procedure TEST(x) which, for a given vertex $x \in V(H)$, checks whether any two graphs H_i, H_j alternate around x. Let us use the term *x-edge* to refer to any edge of H incident to x, and let *x-block* refer to any block of G that contains at least one x-edge.

The procedure TEST(x) proceeds as follows: first, for every x-block G_i, it determines the number of x-edges in G_i and stores this in a counter associated with G_i. This is done by simply looking at every edge incident to x and incrementing the counter of the corresponding block. Next, TEST(x) visits all the x-edges in the order determined by the rotation scheme $\sigma_{\mathcal{H}}(x)$, starting at an arbitrary x-edge. For each x-block it maintains in a counter the number of its x-edges that have been visited so far. An x-block is *active* if some but not all of its x-edges have already been visited.

The procedure TEST(x) also maintains a stack containing the active x-blocks. At the beginning of the procedure the counters of visited edges of each x-block are set to zero and the stack is empty.

For every edge e that TEST(x) visits, it performs the following steps:

1. Let G_i denote the block containing e. Increment the counter of visited x-edges of G_i.
2. If no other edge of G_i has been visited so far, push G_i on the stack.
3. If some x-edge of G_i has been visited before e, we know that G_i is currently somewhere on the stack. Check whether G_i is on the top of the stack. If the top of the stack contains an x-block G_j different from G_i, output that H_i and H_j alternate around x and stop.
4. Check whether e is the last x-edge of G_i to be visited (comparing its counter of visited x-edges to the counter of total x-edges), and if it is, pop G_i from the stack. (Note that if G_i has only one x-edge, it is pushed and popped during the visit of this edge.)

If TEST(x) visits all the x-edges without rejecting, it outputs that there is no alternation around x.

The procedure TEST(x) takes time proportional to the number of x-edges. Thus, we can call TEST(x) for all the vertices $x \in V(H)$ in linear time to test whether there is any alternation in $\mathcal{H}$.

Let us now argue that the procedure TEST(x) is correct. Assume that TEST(x) outputs an alternation of H_i and H_j. This can only happen when G_j is on the top of the stack while an x-edge $e \in G_i$ is visited, and furthermore, e is not the first edge of G_i to be visited. It follows that the first edge of G_i was visited before the first edge of G_j, and G_j is still active when e is visited. This shows that H_i and H_j indeed alternate around x.

Conversely, assume that there is a pair of graphs H_i and H_j that alternate around x, and the alternation is witnessed by two pairs of x-edges $e, e' \in H_i$ and $f, f' \in H_j$. For contradiction, assume that TEST(x) outputs that there is no alternation. Without loss of generality, assume that at least one x-edge of H_i is visited before any x-edge of H_j, that e is visited before

e', and that f is visited before f'. Thus, the four x-edges are visited in the order e, f, e', f'. When the procedure visits e', both G_i and G_j are active, and G_j is on the stack above G_i, since we assumed that the first x-edge of G_i is visited before the first x-edge of G_j. This means that when TEST(x) visited e', G_i was not on the top of the stack and an alternation should have been reported.

This contradiction completes the proof of the lemma. □

LEMMA 4.10. *Let $(G, H, \mathcal{H})$ be an instance of* PEP *where G is connected. Let $G_1, \dots, G_t$ be the blocks of G, and let H_i be the subgraph of H induced by the vertices of G_i. Let $\mathcal{H}_i$ be $\mathcal{H}$ restricted to H_i. Assume that the following conditions hold. 1) each non-trivial H-bridge of G is local, 2) each G_i has an embedding that extends $\mathcal{H}_i$, and 3) no two of the graphs $H_1, \dots, H_t$ alternate around any vertex of H. There is a linear time algorithm which decides whether there exists a facial cycle $\vec{C}$ of $\mathcal{H}$ that separates a pair of vertices x and y such that x and y are connected by a path of G that has no vertex in common with $\vec{C}$.*

Proof. Let P be a path in G with end-vertices in H and let $\vec{C}$ be a facial cycle of $\mathcal{H}$. If P and $\vec{C}$ are vertex-disjoint and the end-vertices of P are separated by $\vec{C}$, we say that P and $\vec{C}$ form a *PC-obstruction.* A PC-obstruction $(P, \vec{C})$ is called *minimal* if no proper subpath $P' \subset P$ forms a PC-obstruction with $\vec{C}$. Observe that, in a minimal PC-obstruction, all the internal vertices of P belong to $V(G) \setminus V(H)$.

We want to show that the existence of a PC-obstruction can be tested in linear time. Of course, it is sufficient to test the existence of a minimal PC-obstruction. Before we explain how this test is done, we make some more observations concerning the structure of minimal PC-obstructions.

Let $(P, \vec{C})$ be a minimal PC-obstruction, and let x and y be the end-vertices of P. As the internal vertices of P belong to $V(G) \setminus V(H)$, then P is a subset of an H-bridge K, and x and y are among the attachments of K. Let us now distinguish two cases, depending on whether K is local to some block or not.

First, assume that K is local to a block B of H. Then, both B and P are part of the same block G_i of G. Hence, $\vec{C}$ belongs to a different block of G, because if it belonged to G_i, then G_i would contain the whole PC-obstruction $(P, \vec{C})$ and it would be impossible to extend the embedding $\mathcal{H}_i$ to G_i, thus contradicting condition 2 of the lemma. Then, let G_j be the block of G that contains $\vec{C}$. Since x and y belong to a common block B of H, they are connected by a path $Q \subseteq B$. Since x and y are separated by $\vec{C}$, Q shares a vertex z with $\vec{C}$ (otherwise the embedding $\mathcal{H}$ would not be planar). Since Q and $\vec{C}$ belong to distinct blocks, z is their unique common vertex. Hence, in the rotation scheme of z, the two edges that belong to Q alternate with the two edges that belong to $\vec{C}$, because $\vec{C}$ separates x and y. Thus, G_i alternates with G_j around z, contradicting condition 3 of the lemma. Then, K cannot be a local bridge.

Second, assume that K is non-local. By condition 1 of the lemma, K consists of a single edge of $E(G) \setminus E(H)$.

We conclude that any minimal PC-obstruction $(P, \vec{C})$ has the property that P is a single edge that forms a non-local H-bridge of G.

Observe that two vertices x and y belonging to distinct blocks of H are separated by a facial cycle of $\mathcal{H}$ if and only if there is no face of $\mathcal{H}$ to which both x and y are incident.

We are now ready to describe the algorithm that determines the existence of a minimal PC-obstruction. The algorithm tests all the edges of $E(G) \setminus E(H)$ one by one. For any such an edge e, it determines in constant time whether it is an H-bridge, i.e., whether its endpoints x and y belong to H. If it is an H-bridge, it checks whether it is non-local in constant time, by using Lemma 4.8. For a non-local bridge, the algorithm then checks in constant time whether there is a face f of H into which this bridge can be embedded, again using Lemma 4.8. Such a face f, if it exists, is uniquely determined. Finally the algorithm checks whether both x and y are incident to f, using the vertex-face incidence graph $\mathcal{VF}$.

Overall, for any edge e, the algorithm determines in constant time whether this edge is a non-local bridge that is part of a minimal PC-obstruction. Thus, in linear time, we determine whether G has any PC-obstruction. □

Combining Lemmata 3.3, 4.8, 4.9 and 4.10 with Theorem 4.3, we obtain the following result.

THEOREM 4.4. PEP *can be solved in linear time when restricted to instances $(G, H, \mathcal{H})$ where G is connected.*

Proof. By Lemma 4.8, an instance of PEP where G is connected can be reduced in linear time to an equivalent instance that has the additional property that all the non-trivial H-bridges are local. Namely, whether an H-bridge K is non-local and, in such a case, which is the face of $\mathcal{H}$ in which K has to be embedded can be computed in time linear in the size of K, by Lemma 4.8. We may thus assume that $(G, H, \mathcal{H})$ is an instance of PEP where G is simply connected and all non-trivial H-bridges in G are local to some block.

To solve PEP for $(G, H, \mathcal{H})$, we present an algorithm based on the characterization of Lemma 3.3. First, we generate all the subinstances $(G_i, H_i, \mathcal{H}_i)$ for $i =$

$1, \ldots, t$, induced by the blocks of G. It is not difficult to see that the subinstances can be generated in linear time. We then solve these subinstances using Algorithm BA, which takes linear time, by Theorem 4.3, since the total size of the subinstances is linear. If any of the subinstances does not have an embedding extension, we reject $(G, H, \mathcal{H})$, otherwise we continue.

In the next step, we check whether there is a pair of graphs H_i, H_j that have an alternation around a vertex of $\mathcal{H}$. If there is an alternation, we reject the instance, otherwise we continue. This step can be implemented in linear time, due to Lemma 4.9.

Finally, we check the existence of PC-obstructions, which by Lemma 4.10 can be done in linear time. We accept the instance if and only if we find no PC-obstruction. The correctness of this algorithm follows from Lemma 3.3. □

Next, we deal with the instances $(G, H, \mathcal{H})$ of Pep in which G is disconnected and H arbitrary. We use Lemma 3.4 directly, and show that the two conditions of the lemma can be checked in linear time. The first condition of Lemma 3.4 can be checked in linear time by Theorem 4.4. To check the second condition, the $\mathcal{CF}$ tree of $\mathcal{H}$ is considered and rooted at any node representing a face; then, the embedding $\mathcal{H}_i$ is considered as $\mathcal{H}$ restricted to the subgraph H_i of H induced by the vertices of G_i; then, for every i, each node of $\mathcal{CF}$ that represents a face of $\mathcal{H}$ incident to a component of H_i and whose parent represents a component of H not in H_i is considered; if there is more than one of such nodes, for some i, $(G, H, \mathcal{H})$ admits no solution, otherwise it does. The correctness of such an argument and an efficient implementation of it are in the proof of the following theorem.

Theorem 4.5. *Pep can be solved in linear time.*

Proof. Let $(G, H, \mathcal{H})$ be an instance of Pep. Let $G_1, \ldots, G_t$ be the connected components of G, let H_i be the subgraph of H induced by the vertices of G_i, and let $\mathcal{H}_i$ be $\mathcal{H}$ restricted to H_i.

By Lemma 3.4, $(G, H, \mathcal{H})$ has an embedding extension if and only if each instance $(G_i, H_i, \mathcal{H}_i)$ has an embedding extension and, for $i \neq j$, no facial cycle of $\mathcal{H}_i$ separates a pair of vertices of H_j. By Theorem 4.4, we can test in linear time whether all the instances $(G_i, H_i, \mathcal{H}_i)$ have an embedding extension.

It remains to test the existence of a facial cycle of $\mathcal{H}_i$ that separates vertices of H_j. For this test, we use the component-face tree $\mathcal{CF}$ of $\mathcal{H}$. Assume that $\mathcal{CF}$ is rooted at any node representing a face of $\mathcal{H}$; call this face the *root face of* $\mathcal{H}$. A face f is an *outer face of* $\mathcal{H}_j$ if at least one child of f in $\mathcal{CF}$ is a component of H_j, but the parent of f does not belong to H_j (which includes the possibility that f is the root face).

We claim that a pair of vertices of H_j is separated by a facial cycle belonging to another component of H if and only if there are at least two distinct outer faces of $\mathcal{H}_j$ in $\mathcal{CF}$. To see this, assume first that $\mathcal{H}_j$ has two distinct outer faces f_1 and f_2, and let C_1 (or C_2) be a component of H_j which is a child of f_1 (or f_2, respectively). Any path from C_1 to C_2 in $\mathcal{CF}$ visits the parent of f_1 or the parent of f_2. These parents correspond to components of H not belonging to H_j, and at least one facial cycle determined by these components separates C_1 from C_2.

Conversely, if C_1 and C_2 are components of H_j separated by a facial cycle belonging to a component C_3 of H_i $(i \neq j)$, then the path in $\mathcal{CF}$ that connects C_1 to C_2 visits C_3, and in such a case it is easy to see that $\mathcal{H}_j$ has at least two outer faces.

We now describe the algorithm that tests the second condition of Lemma 3.4. We assume that each component of H has associated its corresponding subgraph H_i in $\mathcal{CF}$. We then process the components of H one by one and, for each component C, we check whether its parent node is an outer face of the embedding $\mathcal{H}_i$ of the subgraph H_i containing C. We accept $(G, H, \mathcal{H})$ if and only if each $\mathcal{H}_i$ has one outer face. This algorithm clearly runs in linear time. □

The algorithms for Pep we presented in this section are non-constructive, i.e., they test whether an embedding extension exists, without actually constructing the embedding. While it is possible to extend these algorithms into constructive linear-time algorithms, we preferred to present a shorter non-constructive version.

5 Conclusions

In this paper we have shown that Partially Embedded Planarity (Pep), i.e. the problem of deciding whether a partial drawing can be extended to a planar drawing of the entire graph, is solvable in linear time.

The following two generalizations of Pep are NP-complete since they are special cases of Crossing number and Maximum planar subgraph, respectively: (i) deciding if an embedding $\mathcal{H}$ can be extended to a planar drawing of G with at most k crossings; and (ii) deciding if at least k edges of $E(G) \setminus E(H)$ can be added to $\mathcal{H}$ preserving planarity.

Two additional problems that generalize Pep in different directions are the following: (iii) deciding whether G has a planar embedding $\mathcal{G}$ in which at least k edges of H are embedded the same as in $\mathcal{H}$; and (iv) deciding whether a set F of at most k edges can be deleted from H, so that $G \setminus F$ has a planar embedding $\mathcal{G}$

in which the induced embedding of $H \setminus F$ coincides with $\mathcal{H} \setminus F$. It can be proved that even these two problems, called Minimum Rerouting Partially Embedded Planarity and Maximum Preserved Partially Embedded Planarity, respectively, are NP-hard.

The results presented in this paper yield to solve in linear time the problem of deciding whether two graphs have a Simultaneous Embedding with Fixed Edges [7, 12, 13, 5, 8, 11, 10] (in the following called SEFE, for short) if one of the graphs has a fixed embedding. A SEFE of a pair of graphs $(G_1 = (V, E_1), G_2 = (V, E_2))$ on the same set of n vertices is a pair of planar drawings (Γ_1, Γ_2) of (G_1, G_2) such that each edge $(u, v) \in E_1 \cap E_2$ has the same drawing in Γ_1 and in Γ_2. The following theorem immediately implies a linear-time algorithm for deciding whether two graphs have a SEFE, if one of them has a fixed embedding.

Theorem 5.1. *Let G_1 and G_2 be two graphs with the same n vertices, let $\mathcal{G}_2$ be a planar embedding of G_2 and let $\mathcal{G}_{1\cap 2}$ be the restriction of $\mathcal{G}_2$ to $G_1 \cap G_2$. Then G_1 and G_2 have a SEFE in which the planar embedding of G_2 is $\mathcal{G}_2$ if and only if $(G_1, G_1 \cap G_2, \mathcal{G}_{1\cap 2})$ is a* Yes-*instance of* Pep.

References

[1] P. Bertolazzi, G. Di Battista, and W. Didimo. Computing orthogonal drawings with the minimum number of bends. *IEEE Trans. Computers*, 49:8:826–840, 2000.

[2] J. M. Boyer and W. J. Myrvold. On the cutting edge: simplified $O(n)$ planarity by edge addition. *J. Graph Alg. Appl.*, 8(3):241–273, 2004.

[3] H. de Fraysseix, P. O. de Mendez, and P. Rosenstiehl. Trémaux trees and planarity. *Int. J. Found. Comput. Sci.*, 17:1017–1030, 2006.

[4] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25:956–997, 1996.

[5] E. Di Giacomo and G. Liotta. Simultaneous embedding of outerplanar graphs, paths, and cycles. *Int. J. Comput. Geom. Appl.*, 17(2):139–160, 2007.

[6] Christoph Dornheim. Planar graphs with topological constraints. *J. Graph Alg. Appl.*, 6(1):27–66, 2002.

[7] C. Erten and S. G. Kobourov. Simultaneous embedding of planar graphs with few bends. In J. Pach, editor, *GD '04*, volume 3383 of *LNCS*, pages 195–205, 2004.

[8] A. Estrella-Balderrama, E. Gassner, M. Jünger, M. Percan, M. Schaefer, and M. Schulz. Simultaneous geometric graph embeddings. In S.-H. Hong, T. Nishizeki, and W. Quan, editors, *GD '07*, volume 4875 of *LNCS*, pages 280–290, 2007.

[9] J. Fiala. NP-completeness of the edge precoloring extension problem on bipartite graphs. *J. Graph Theory*, 43(2):156–160, 2003.

[10] J. Fowler, C. Gutwenger, M. Jünger, P. Mutzel, and M. Schulz. An SPQR-tree approach to decide special cases of simultaneous embedding with fixed edges. In I. G. Tollis and M. Patrignani, editors, *GD '08*, volume 5417 of *LNCS*, pages 157–168, 2008.

[11] J. Fowler, M. Jünger, S. G. Kobourov, and M. Schulz. Characterizations of restricted pairs of planar graphs allowing simultaneous embedding with fixed edges. In H. Broersma, T. Erlebach, T. Friedetzky, and D. Paulusma, editors, *WG '08*, volume 5344 of *LNCS*, pages 146–158, 2008.

[12] F. Frati. Embedding graphs simultaneously with fixed edges. In M. Kaufmann and D. Wagner, editors, *GD '06*, volume 4372 of *LNCS*, pages 108–113, 2006.

[13] E. Gassner, M. Jünger, M. Percan, M. Schaefer, and M. Schulz. Simultaneous graph embeddings with fixed edges. In F. V. Fomin, editor, *WG '06*, volume 4271 of *LNCS*, pages 325–335, 2006.

[14] C. Gutwenger, K. Klein, and P. Mutzel. Planarity testing and optimal edge insertion with embedding constraints. *J. Graph Alg. Appl.*, 12(1):73–95, 2008.

[15] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In J. Marks, editor, *GD '99*, volume 1984 of *LNCS*, pages 77–90, 2000.

[16] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4), 1974.

[17] M. Juvan and B. Mohar. 2-restricted extensions of partial embeddings of graphs. *European J. Comb.*, 26(3–4):339–375, 2005.

[18] L. Kowalik and M. Kurowski. Short path queries in planar graphs in constant time. In *STOC '03*, pages 143–148, 2003.

[19] J. Kratochvil and A. Sebo. Coloring precolored perfect graphs. *J. Graph Theory*, 25:207–215, 1997.

[20] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15:271–283, 1930.

[21] B. Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discr. Math.*, 12(1):6–26, 1999.

[22] P. Mutzel. The SPQR-tree data structure in graph drawing. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *ICALP '03*, volume 2719 of *LNCS*, pages 34–46, 2003.

[23] M. Patrignani. On extending a partial straight-line drawing. *Found. Comput. Sci.*, 17(5):1061–1069, 2006.

[24] J. A. La Poutré. Alpha-algorithms for incremental planarity testing. In *STOC '94*, pages 706–715, 1994.

[25] R. Tamassia. On-line planar graph embedding. *J. Algorithms*, 21(2):201–239, 1996.

[26] R. Tamassia. Constraints in graph drawing algorithms. *Constraints*, 3(1):87–120, 1998.

[27] R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst., Man and Cyber.*, (1):61–79, 1988.

[28] J. Westbrook. Fast incremental planarity testing. In W. Kuich, editor, *ICALP '92*, volume 623 of *LNCS*, pages 342–353, 1992.