Roma Tre University
Ph.D. in Computer Science and Engineering

On the Existence and Optimality of Some Planar Graph Embeddings

Patrizio Angelini

On the Existence and Optimality of Some Planar Graph Embeddings

A thesis presented by
Patrizio Angelini
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Engineering

Roma Tre University
Dept. of Informatics and Automation

April 2010

Committee:
Prof. Giuseppe Di Battista

Reviewers:
Prof. Anna Lubiw
Prof. Seok-Hee Hong

Ai miei fratelli...e ar roscio!

When the solution is simple, God is answering. Where the world ceases to be the scene of our personal hopes and wishes, where we face it as free beings admiring, asking and observing, there we enter the realm of Art and Science. (A. Einstein)

Acknowledgments

My best acknowledgments go to my advisor Giuseppe Di Battista. Knowing him and working with him during my undergraduate studies was an event that changed my life, as it was the first step towards the direction I am following now. I thank him for having taught me how to make research and for having transmitted to me his passion in doing it.

Special thanks go to Fabrizio Brillo Frati, he was not just a research mate for me, but he was my second advisor, he is not just a guy I spent most of my time with in the last three years, but he is a true friend.

I would like to also thank Maurizio Titto Patrignani, a key element of the great atmosphere we have in our group.

I would like to thank Michael Kaufmann for allowing me to spend four wonderful months as a guest at the University of Tübingen, and Jan Kratochvíl, Michael Goodrich, and David Eppstein for the very pleasant and interesting visits I could make at the Charles University of Prague and at the University of California, Irvine.

I would like to thank Anna Lubiw and Seok-Hee Hong for reviewing this thesis.

I would like to thank my coauthors for the great time I had when working with all of them: Luca Cittadini, Pier Francesco Cortese, Giuseppe Di Battista, Walter Didimo, Fabrizio Frati, Markus Geyer, Luca Grilli, Vít Jelínek, Michael Kaufmann, Jan Kratochvíl, Daniel Neuwirth, Maurizio Patrignani, Ignaz Rutter, and Antonios Symvonis.

I thank all the other people that made part of our research group during my Ph.D. years, as each of them contributed to make it a great period: Guido Drovandi, Fabrizio Martorelli, Alessandro Marzioni, Bernardo Palazzi, Maurizio Pizzonia, Tiziana Refice, Massimo Rimondini, and Stefano Vissicchio.

The final heartfelt acknowledge is for my friends and for my family. Without them and their love, everything else would be pointless to me.

Contents

Introduction

A *graph* is an abstract mathematical representation of a set of objects, called *vertices*, together with a pairwise relationship between such objects, that is represented by a collection of *edges* connecting pairs of vertices. Examples of relationships among objects that are representable by a graph can be found in every field, ranging from interpersonal relationships to computer networks and from knowledge representation to bioinformatics. Of course, the best way to make such a relationship clearly understandable is to visualize the graph so that vertices and edges are easily recognizable at human eye. Such an issue is addressed in the research field of *Graph Drawing*, which can be regarded as a cross between the areas of *Graph Theory*, *Graph Algorithmic*, and *Computational Geometry*.

In Graph Drawing, the most common way to visualize a graph is to draw each vertex as a point in the plane and each edge as a curve connecting the corresponding points. The placement of the vertices in the plane and the drawing of the curves should be performed in such a way that the resulting drawing be nice and readable, in the sense that the information described by the graph should be possibly understandable at a glance. In order to obtain the desired nice and readable visualization, it is important to formalize the aesthetic criteria that distinguish a "good" drawing from a "bad" one. Then, the goal of Graph Drawing is to create algorithms that automatically produce drawings respecting such criteria.

The most natural aesthetic criterion that one can think of is probably the absence of partial or complete overlapping among vertices and edges, that is called *planarity*. Another important criterion that one has to consider when drawing a graph is the *area* of the drawing, as a drawing with a small area can be better visualized inside a small screen. Observe that, while planarity is a *property* that a drawing may satisfy or not, the drawing area is a *measure of quality* that can be used to compare two drawings. Many other properties and

quality measures can be defined concerning the visualization of graphs, even with the possibility of combining some of them. However, the "best" drawing of a graph might not exist, since drawings that are "good" in terms of a certain criterium may be "bad" in terms of another.

It is interesting to observe that some of the aesthetic criteria of a drawing of a graph only depend on its topological features, while other criteria also depend on the geometrical realization. For example, an important theorem in Graph Drawing, known as *Fary's Theorem*, states that every graph admitting a planar drawing also admits a planar drawing in which edges are represented by straight-line segments. This implies that, in order to decide whether a given graph admits a planar drawing, it is possible to study whether such a graph admits a *planar embedding*, that is, a circular ordering of the edges around each vertex that determines no topological crossing in the induced drawing, rather than computing the actual coordinates of the points representing the vertices. On the other hand, given a graph with a fixed embedding, different geometrical realizations may lead to drawings with different area.

Several natural and interesting to study questions can be formulated concerning the aesthetic criteria defined for the graphical representation of graphs.

When considering a graph property, the first, and probably most natural, arising question is the one about the existence of graphs satisfying such a property and of graphs not satisfying it. If both positive and negative instances exist, the problem can then be studied either from a combinatorial or from an algorithmic point of view. In the former case, one can ask for a relationship between the satisfiability of such a property and the structure of the graph. Namely, it is interesting to understand whether there exists a family of graphs such that all and only the graphs belonging to it satisfy the desired property. In the latter case, one can ask for the computational complexity of the problem of deciding whether a given graph satisfies the property.

Question 1 *Given a property P, characterize the family of graphs $\mathcal{F}$ such that a graph G admits a drawing (an embedding) satisfying P if and only if $G \in \mathcal{F}$.*

Question 2 *Given a graph G and a property P, what is the time complexity of deciding whether G satisfies P? Also, what is the time complexity of computing the drawing (the embedding) of G satisfying P?*

Explanatory examples for the two problems come from the study of graph planarity. Regarding Question 1, a fundamental theorem on graph planarity, known as *Kuratowsky's theorem*, states that a graph is planar if and only if it

does not contain any subgraph that is a subdivision of the complete graph K_5 or of the complete bipartite graph $K_{3,3}$. Concerning Question 2, several linear-time planarity testing algorithms are known in the literature (see Sect. 1.2 for a brief overview about planarity), so as some NP-hardness proofs related to graph planarity, like the one concerning the problem of finding the maximal planar subgraph of a non-planar graph. Of course, the two problems are often strictly related, since a combinatorial characterization for a certain property may directly lead to a polynomial-time algorithm for testing it.

On the other hand, when considering a particular measure of quality of a drawing or of an embedding, the two most natural questions are certainly the one asking for the optimal value of such a measure among all the graphs of a certain family and the one asking for an efficient algorithm that optimizes such a measure for a given graph.

Question 3 *Given a measure M and a family of graphs $\mathcal{F}$, what is the optimal value of M among all the graphs $G \in \mathcal{F}$?*

Question 4 *Given a graph G and a measure M, what is the time complexity of computing a drawing (an embedding) of G that is optimal with respect to M?*

For these two questions, clear examples can be found in the context of the area-minimization problem. In fact, many results are known in Graph Drawing about lower-bounds and upper-bounds for the area needed to draw several families of graphs under certain constraints, and such upper-bounds are generally obtained as a result of efficient drawing algorithms.

In this thesis we address and partially answer Questions 1–4 on several classes and types of graphs. We mainly deal with planar graphs and with graph properties and measures that depend on the topology of the graph rather than on its geometry. We propose algorithms for computing planar embeddings or drawings that satisfy certain properties or that are optimal with respect to certain measures of quality. Also, we consider the same questions on *simultaneous graph drawing* problems, that is, problems involving more than one graph, and on *clustered graphs*, that is, graphs where the vertices are grouped into clusters by means of a hierarchical structure.

Part I of this thesis deals with planar graphs and with the most common methods to describe and handle their planar embeddings.

In Chapter 1 we introduce some preliminaries and definitions about planar graphs, their embeddings, and their drawings.

In Chapter 2 we introduce basic concepts about the connectivity of graphs and we illustrate the main techniques for describing and handling the embeddings of planar graphs, depending on their degree of connectivity. In particular, we consider *SPQR-trees*, a data-structure describing the decomposition of a biconnected graph into its triconnected components.

Part II deals with problems related to properties and quality measures of planar graphs that only depend on their embedding.

In Chapter 3 we consider the problem of comparing two different embeddings of the same graph in terms of the minimum number of elementary operations that have to be performed in order to turn one embedding into the other. For this problem, that we call *Topological Morphing*, we define the basic operations that can be performed to transform an embedding, we show that the problem is NP-hard for biconnected planar graphs, and we present polynomial-time algorithms for some more constrained cases that induce a fixed-parameter tractable algorithm for the general case.

In Chapter 4 we introduce and solve a problem related to the classical planarity testing and again related to the comparison between embeddings. In fact, given a planar embedding of a subgraph of a planar graph, we consider the problem of extending it to a planar embedding of the whole graph while maintaining the embedding of the subgraph unchanged. Although many polynomial-time solvable problems become harder when a partial solution is fixed in advance, we show that this is not the case for planarity, as we describe a combinatorial characterization of the planar graphs that admit an embedding extension and we present a linear-time testing algorithm.

In Chapter 5 we deal with the minimization of certain distance measures that describe the quality of a planar embedding. In particular, we study the problem of finding a *minimum-depth embedding* of a planar graph, where the *depth* of an embedding is the maximum distance of its internal faces from the external one. We present an $O(n^4)$-time algorithm, improving the best previous bound, that was $O(n^5 \log n)$, obtained by Bienstock and Monma [BM90].

In Part III of this thesis we deal with *greedy drawings* of graphs, i.e., drawings of graphs satisfying a particular property that is defined in terms of geometrical distances between vertices and is related to the effectiveness of *greedy routing* algorithms for sensor networks.

In Chapter 6 we give preliminary definitions on greedy drawings and we show that every triconnected planar graph admits a greedy drawing, hence proving a conjecture by Papadimitriou and Ratajczac.

In Chapter 7 we consider area requirements of greedy drawings and we prove that such requirements can be non-polynomial in the worst-case, as we present

greedy-drawable trees requiring exponential area in any greedy drawing.

Part IV deals with the simultaneous embedding of two graphs on the same set of points on the plane.

In Chapter 8 we deal with the *geometric simultaneous embedding* problem, in which both graphs are required to be drawn with straight-line edges. We show that there exist a tree and a path that do not admit any of such embeddings. This problem was the main open problem in the area, as it fills the gap in the combinatorial characterization of the pairs of graphs always admitting a geometric simultaneous embedding. In fact, it was already known that two caterpillars always admit such embedding, while there exist two trees not admitting any.

Part V deals with clustered graphs, a type of graphs in which the vertices are grouped into sets, called *clusters*, according to a certain given hierarchy.

In Chapter 9, we give some preliminaries and definitions about clustered graphs and the problem of deciding whether a clustered graph admits a *c-planar drawing*, that is, a planar drawing in which some planarity constraints are required also for the graphical representation of the clusters. The complexity of such a problem is unknown in the general case, while there exist characterizations and efficient algorithms for some particular classes of graphs. Also, we define a generalization of the problem, in which clusters can be split in order to make the clustered graph c-planar.

In Chapter 10 we deal with the problem of drawing clustered graphs nicely. In this context, we show that every c-planar clustered graph admits a straight-line c-planar drawing in which clusters are represented by axis-parallel rectangles. As the same result holds for any convex shape fixed in advance, we can state that the c-planarity of clustered graphs is independent on the geometrical representation of the clusters, so as Fary's Theorem states that the planarity of graphs is independent on the geometrical representation of the edges.

Part I

Preliminaries

Chapter 1

Graph Preliminaries and Definitions

In this chapter, we introduce some preliminaries and definitions about graphs, their embeddings, and their drawings.

A reader who wants to assume more familiarity with the basic concepts about graphs, algorithms, and geometry, may refer to books on Graph Theory (e.g., [Har72, BM76, NC88, Die05]), to books on Algorithms (e.g., [Eve79, AHU83, CLRS01, GT02]), and to books on Computational Geometry (e.g., [PS85, Ede87, dvKOS00]). Reference books containing detailed definitions, basic concepts, and most important results about Graph Drawing can also be useful and interesting to read [DETT99, KW01, NR04].

The chapter is organized as follows. In Sect. 1.1 we give basic definitions about graphs. In Sect. 1.2 we deal with planar graphs and planar embeddings; in Sect. 1.3 we describe the main drawing conventions used in Graph Drawing; finally, in Sect. 1.4 we define some interesting subclasses of planar graphs.

1.1 Basic Definitions

A *graph* $G = (V, E)$ is composed of a set V of *vertices* or *nodes*, and a multiset E of *unordered* pairs of vertices, called *edges* or *arcs*. If the pairs of vertices in E are ordered, G is a *directed graph*, also referred to as *digraph*. The graph obtained from a digraph G by considering its edges without orientation is called the *underlying graph* of G. Given an edge $e = (u, v) \in E$, we say that u and v are *incident* to e (u and v are the *end-vertices* of e), and that e is *incident* to

u and v. Also, we say that two vertices are *adjacent* if they are incident to the same edge, and two edges are *adjacent* if they are incident to the same vertex.

A *self-loop* in a graph $G = (V, E)$ is an edge $(u, u) \in E$. A set of *multiple edges* in a graph (V, E) is a set of edges connecting the same two vertices $u, v \in V$. A graph is *simple* if it contains neither self-loops nor multiple edges. In the following, unless otherwise specified, we always refer to simple graphs.

A graph $G = (V, E)$ is *complete* if for each pair of vertices $u, v \in V$, edge $(u, v) \in E$. A graph is *bipartite* if V can be partitioned into two sets V_1 and V_2 such that for each edge $(u, v) \in E$, either $u \in V_1$ and $v \in V_2$ or vice versa. A graph $G = (V, E)$ is *bipartite* if its vertex set V can be partitioned into two subsets V_1 and V_2 so that every edge of E is incident to a vertex of V_1 and to a vertex of V_2.

The *degree of a vertex* is the number of edges incident to it. The *degree of a graph* is the maximum among the degrees of its vertices.

A graph $G' = (V', E')$ is a *subgraph* of a graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. We say that $G' = (V', E')$ is *induced* by V' if, for each edge $(u, v) \in E$ such that $u, v \in V'$, we have $(u, v) \in E'$. Also, $G' = (V', E')$ is a *spanning subgraph* of $G = (V, E)$ if it is a subgraph of G and $V' = V$.

A *subdivision* of a graph G is a graph G' that can be obtained by replacing each edge of G with a path of arbitrary length. The *contraction* of an edge (u, v) consists of the replacement of u, v, and (u, v) with a single vertex w, of each edge (u, z) with an edge (w, z), and of each edge (v, z') with an edge (w, z'). A *minor* of a graph G is any graph that can be obtained from G by a sequence of removals of vertices, removals of edges, and contractions of edges.

1.2 Planar Graphs

In this section we give preliminaries and definitions about planar graphs.

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 open Jordan curve between the points to which the end-vertices of the edge have been mapped. A drawing is *planar* if no two edges intersect except, possibly, at common end-points. A *planar graph* is a graph admitting a planar drawing. A planar drawing of a graph determines a circular ordering of the edges incident to each vertex v, that we call the *rotation scheme* of v. Two drawings of the same graph are *equivalent* if they determine the same rotation scheme for each vertex. Fig. 1.1 shows two equivalent drawings of a planar graph. A *combinatorial embedding* (or simply *embedding*) is an equivalence class of planar drawings. A planar drawing, or

equivalently a combinatorial embedding, partitions the plane into topologically connected regions called *faces*. A vertex (an edge) is *incident* to a face if it belongs to the cycle delimiting the face. All the faces are bounded by the cycle delimiting them, except for one face, that we call *outer face* (or *external face*). The other faces are called *internal faces*. A *planar embedding* $\langle \Gamma, f \rangle$ of a graph G is composed of an embedding Γ of G and of an outer face f of Γ. A *plane graph* is a graph with a fixed planar embedding.

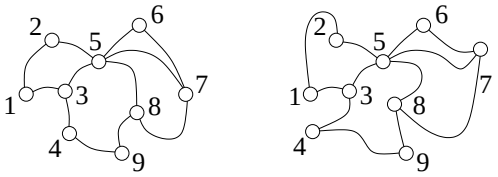


Figure 1.1: Two equivalent planar drawings of a planar graph.

A plane graph is *maximal* (or equivalently is a *triangulation*) if all its faces are delimited by cycles of three vertices. A planar graph is *maximal* if it admits a planar embedding that determines a triangulation. A triangulation is maximal in the sense that adding an edge to it yields a non-planar graph. Maximal planar graphs are an important and deeply studied class of planar graphs since any planar graph can be augmented to maximal by adding *dummy edges* to it and since triangulations, as the triconnected planar graphs, admit exactly one combinatorial embedding (a more detailed discussion about the properties of highly-connected planar graphs is given in Sect. 2.1) and hence are often easier to deal with. A plane graph is *internally-triangulated* when all its internal faces have exactly three incident vertices.

The *dual graph* of an embedded planar graph G has a vertex for each face of G and an edge (f, g) for each two faces f and g of G sharing an edge. Fig. 1.2 shows an embedded planar graph and its dual graph. The dual graph of G only depends on the embedding of G and not on the choice of the external face.

Planarity

Planarity is commonly accepted as the most important aesthetic criteria a drawing should satisfy to be nice and readable. In fact, the absence of partial or complete overlapping among the objects makes the drawing aesthetically pleasant and easily readable by the human eye, and provides extremely high read-

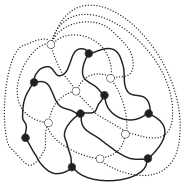


Figure 1.2: A planar graph, whose vertices are represented by black circles and whose edges are represented by thick lines, and its dual graph, whose vertices are represented by white circles and whose edges are represented by dotted lines.

ability of the combinatorial structure of the graph, as confirmed by some cognitive experiments in graph visualization [PCJ97, Pur00, PCA02, WPCM02]. See Fig. 1.3 for a comparison between a non-planar and a planar drawing. However, the great importance of planar graphs, so in Graph Drawing as in Graph Theory and Computational Geometry in general, also comes from the many mathematical, combinatorial, and geometrical properties they exhibit.

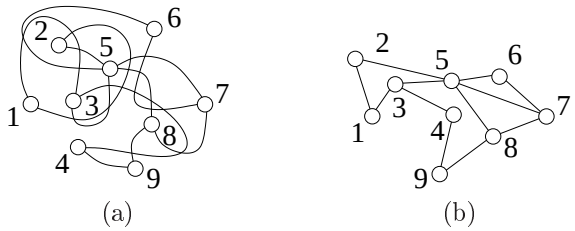


Figure 1.3: (a) A non-planar drawing of a planar graph G. (b) A planar drawing of G.

From the combinatorial and topological point of view, the first important result about planar graphs is the characterization given by Kuratowski [Kur30] in 1930, stating that a graph is planar if and only if it contains no subdivision of the complete graph K_5 with five vertices and no subdivision of the complete bipartite graph $K_{3,3}$ with three vertices in each of the sets of the bipartition. Such a characterization has been extended by Wagner, who stated that a graph is planar if and only if it contains no K_5-minor and no $K_{3,3}$-minor [Wag37]. The planarity of a graph can be tested in linear time, as first shown by Hopcroft and Tarjan [HT74] in 1974. Linear-time algorithms for testing the planarity of a

graph are also presented, e.g., in [BL76, ET76, dR82, BM04, dFdMR06, HT08]. Also, such testing algorithms can be suitably modified in order to compute planar embeddings in the case the test is positive. If an embedding of a graph is fixed, then linear time still suffices to test if the embedding is planar [Kir88]. The fact that the planarity testing, so as many other problems on planar graphs, can be solved in linear time is due to another important mathematical property of planar graphs, stating that the number of edges of a planar graph is linear in the number of its vertices. Namely, by the Euler's formula, we have $m \leq 3n-6$, where m is the number of edges, in any n-vertex planar graph.

1.3 Drawing Conventions

When aiming at high readability of a drawing, another important issue that has to be considered concerns the geometrical representation of the edges and of the faces. Namely, planar drawings in which edges are represented by straight-line segments (known as *straight-line drawings*, see Figs. 1.4(a) and (c)) happen to be more readable than drawings in which edges are represented by poly-lines (known as *poly-line drawings*, see Fig. 1.4(b)) or general curves, and drawings in which faces are drawn as convex polygons (known as *convex drawings*, see Fig. 1.4(c)) are more readable than drawings in which this is not the case (see Fig. 1.4(a)). Among the more used and studied drawing conventions, we also mention *orthogonal drawings*, in which each edge is represented by a sequence of horizontal and vertical segments.

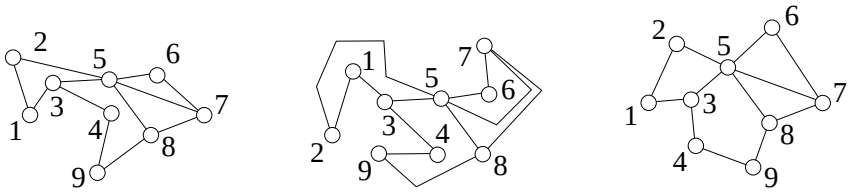


Figure 1.4: (a) A straight-line planar drawing of a planar graph G. (b) A poly-line planar drawing of G. (c) A convex drawing of G.

Other drawing conventions that are worth to mention are the *grid drawings*, in which vertices and bends have integer coordinates, *upward drawings* of digraphs, in which each edge is represented by a curve monotonically-increasing in the upward direction, and *proximity drawings*, in which given a definition

of *proximity*, the *proximity graph* of a set of points is the graph with a vertex for each point of the set, and with an edge between two vertices if the corresponding points satisfy the proximity property. Then, a proximity drawing of a graph G is a drawing D of G such that the proximity graph of the set of points on which the vertices of G are drawn in D coincides with G itself. An example of proximity graphs is the *Delaunay triangulation* for a set P of points in the plane, that is, a triangulation T such that no point in P is inside the circumscribed circle of any triangle in T.

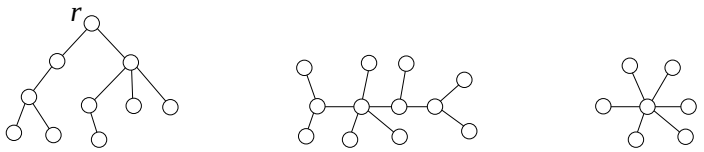
The most studied and used drawing convention is the one of straight-line drawings. Of course such a convention is much more restrictive than the one in which edges can have bends, and hence many results that hold for poly-line drawings do not hold for straight-line drawings. However, regarding planarity, this is not the case. Indeed, a very important result, known as Fary's theorem and independently proved by Wagner [Wag36], by Fary [Far48], and by Stein [Ste51], states that a graph admits a straight-line planar drawing if and only if it admits a planar drawing. This result shows that planarity does not depend on the geometry used for representing the edges but it only depends on the topological properties of the graph.

Some aesthetic criteria can be defined to measure the quality of a drawing. Among them, one of the most important is certainly the area occupied by the drawing, that is, the area of the smallest rectangle with sides parallel to the coordinate axes that contains all the drawing. Of course, small area drawings can not be obtained by simply scaling down the drawing, since some *resolution rules* have to be respected in the drawing for maintaining readability. In particular, a minimum distance, say one unit, between two elements (vertices and edges) of the drawing has to be maintained. In order to respect some of such rules, when dealing with area minimization problems, vertices are usually placed on an integer grid, in such a way that the minimum distance between any two of them is at least one grid unit. In this direction, it has been shown in several papers that every n-vertex plane graph admits a planar straight-line drawing on a $O(n^2)$ area grid [dPP88, dPP90, Sch90, CN98, ZH03, BFM07]. Further, a grid of quadratic size is asymptotically the best possible for straight-line planar drawings, since there exist planar graphs requiring such an area in any planar grid drawing [Val81, dPP90, FP07].

1.4 Families of Planar Graphs

In this section we present preliminaries and definitions about some important sub-classes of planar graphs that will be used in the rest of the thesis.

A *cycle* is a connected graph such that each vertex has degree exactly two. A *tree* is a connected acyclic (i.e., not containing any cycle) graph (see Fig. 1.5(a)). A *path* is a tree such that each vertex has degree at most two. A *chord* of a cycle (of a path) is an edge connecting two non-consecutive vertices of the cycle (of the path). A *leaf* of a tree is a node of degree one. A *leaf edge* is an edge incident to a leaf. A *caterpillar* (see Fig. 1.5(b)) is a tree such that removing all the leaves and all the leaf edges yields a path, called *spine* of the caterpillar, whose nodes and edges are called *spine nodes* and *spine edges*, respectively. A *star graph* (see Fig. 1.5(c)) is a tree such that removing all the leaves and all the leaf edges yields an isolated node, called *central node*.



(a) (b) (c)

Figure 1.5: (a) A tree rooted at a node r. (b) A caterpillar. (c) A star graph.

A *rooted tree* is a tree with one distinguished node, called *root*. In a rooted tree, the *depth* of a node v is the length of the unique path (i.e., the number of edges composing the path) between v and the root. The *depth* of a rooted tree is the maximum depth among all the vertices.

A *binary tree* (a *ternary tree*) is a rooted tree such that each node has at most two children (resp. three children). A tree is *ordered* if an order of the children of each node (i.e., a planar embedding) is specified. In an ordered binary tree we distinguish the *left* and the *right child* of a node. The *subtrees* of a node u of a tree T are the subtrees of T rooted at u and not containing the root of T.

An *outerplanar graph* is a graph admitting an *outerplanar embedding*, that is, an embedding in which all the vertices are on the outer face. From a combinatorial point of view, an outerplanar graph is a graph that contains no K_4-minor and no $K_{2,3}$-minor (see Fig. 1.6 (a)).

A *series-parallel graph* is inductively defined as follows. An edge (u, v) is

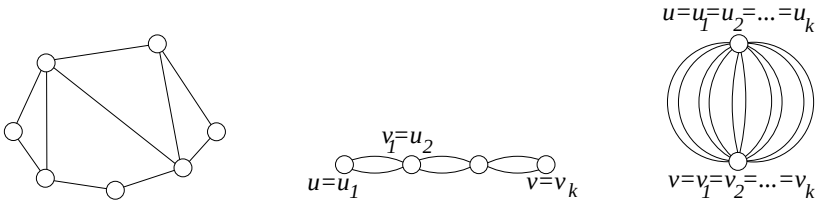


Figure 1.6: (a) An outerplanar graph. (b) A series composition of a sequence $G_1, G_2, \ldots, G_k$ of series-parallel graphs. (c) A parallel composition of a set $G_1, G_2, \ldots, G_k$ of series-parallel graphs.

a series-parallel graph with *poles* u and v. Denote by u_i and v_i the poles of a series-parallel graph G_i. A *series composition* of a sequence $G_1, \ldots, G_k$ of series-parallel graphs, with $k \geq 2$, is a series-parallel graph with poles $u = u_1$ and $v = v_k$ such that v_i and u_{i+1} have been identified, for each $i = 1, \ldots, k-1$ (see Fig. 1.6 (b)). A *parallel composition* of a set $G_1, \ldots, G_k$ of series-parallel graphs, with $k \geq 2$, is a series-parallel graph with poles $u = u_1 = \cdots = u_k$ and $v = v_1 = \cdots = v_k$ (see Fig. 1.6 (c)). From a combinatorial point of view, a series-parallel graph is a graph that contains no K_4-minor.

A *Hamiltonian cycle (path)* in a graph G is a simple cycle (resp. path) passing through all the vertices of G. A *Hamiltonian graph* is a graph containing a Hamiltonian cycle.

Chapter 2

Decomposition of Planar Graphs

In this chapter we study the problem of decomposing planar graphs into components with higher degree of connectivity and we present the main data structures that can be used to deal with such decomposition and to describe and handle the embeddings of planar graphs. In particular, we describe *SPQR-trees*, a data structure that can be used to represent and to efficiently handle all the embeddings of a biconnected planar graph.

The structure of the chapter is as follows. In Sect. 2.1 we give some basic definitions about connectivity and about some properties that are depending on the degree of connectivity, while in Sect. 2.2 we define the main data structures to describe the decomposition into biconnected and triconnected components, with particular focus on the SPQR-trees.

2.1 Connectivity

Let $G = (V, E)$ be a planar graph. We say that G is *connected* if, for each pair of vertices $u, v \in V$, there exists a path connecting u and v. More generally, we say that G is *k-connected* if, for each pair of vertices $u, v \in V$, there exist k disjoint paths connecting u and v. Alternatively, we say that G is k-connected if 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-sets and separating 2-sets are also 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*, while the maximal triconnected subgraphs of a graph are its *triconnected components*. Each edge of G falls into a single block of G and into a single triconnected component, while cutvertices (resp., vertices belonging to a separation pair) are shared by different blocks (resp., by different separation pairs). A *cut* of a graph $G = (V, E)$ is a partition if its vertices into two subsets V_1 and V_2. A *cutset* of G is the set $E' \subseteq E$ such that $(u, v) \in E'$ if and only if $u \in V_1$ and $v \in V_2$. Observe that the removal of the edges of E' from E increases the number of connected components of G.

Connectivity and Embeddings

In this subsection we give some properties of planar graphs and planar embeddings that depend on the degree of connectivity of the graph.

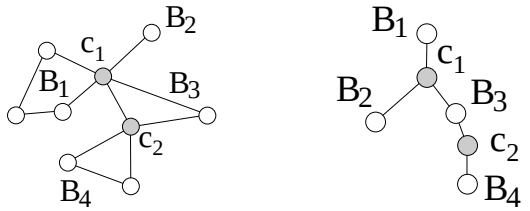
First, observe that every 3-connected planar graph admits two planar embeddings, which only differ for a flipping around their poles, that is, the list of incident edges around each vertex in the two embeddings are one the reversal of the other. Hence, problems concerning the research of a particular embedding of a planar graph that are difficult for general planar graphs, can be efficiently solved when the graph is triconnected. This property strongly motivates the study of problems on low-connected planar graphs in terms of their decomposition into highly-connected components, as the problem can be solved more easily on such components and then the problems turns into the one of putting them together efficiently. Examples of such problems are the subject of the chapters of the next part of this thesis.

Another important property regarding highly connected graphs is that every 4-connected planar graph is Hamiltonian [Tut56] and the Hamiltonian cycle can be found efficiently. Observe that the problem of finding a Hamiltonian cycle in a general planar graph, even if triconnected, is NP-hard [GJ79]. Further, every biconnected outerplanar graph G has exactly one Hamiltonian cycle, namely the one delimiting the face to which all the vertices of G are incident in an outerplanar embedding of G.

2.2 Data Structures for Planar Graphs

In order to describe and efficiently handle the decomposition of a connected graph into biconnected components and of a biconnected graph into triconnected components, some efficient data structures have been defined.

The data structure that can be used to describe the decomposition of a connected graph into its biconnected components, called *block-cutvertex tree* (usually referred to as *BC-tree*), was introduced by Harary and Prins in [HP66]. The BC-tree $\mathcal{T}$ of a connected graph G is a tree containing a B-node for each block of G and a C-node for each cutvertex of G. Edges in $\mathcal{T}$ connect each B-node μ to the C-nodes associated with the cutvertices belonging to the block of μ. The BC-tree of G may be thought as rooted at a specific block ν. The number of nodes of $\mathcal{T}$ is equal to the number of blocks plus the number of cutvertices, that is $O(n)$, where n is the number of vertices of G. Fig. 2.1 shows a connected planar graph and its block-cutvertex tree, rooted at a block B_1.



(a) (b)

Figure 2.1: (a) A connected planar graph and (b) its block-cutvertex tree, rooted at B_1.

The data structure that can be used to describe the decomposition of a biconnected graph into its triconnected components, called *SPQR-tree*, was introduced by Di Battista and Tamassia in [DT90, DT96b, DT96a]. In the following we define SPQR-trees, we give their main properties, and we describe how such trees can be used to represent and efficiently handle all the embeddings of a planar biconnected graph.

SPQR-Trees

SPQR-trees were introduced in [DT96b] to describe all the possible embeddings of biconnected planar graphs in a succinct way and were used in various situations when asking for planar embeddings with special properties.

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 an 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'. A vertex $w \neq u, v$ belongs to exactly one maximal split component of $\{u, v\}$. We call *split component* of $\{u, v\}$ the union of any number of maximal split components of $\{u, v\}$.

In [DT96b], SPQR-trees were introduced as rooted at one edge of G, called *reference edge*. However, they can also be viewed as unrooted, since a decomposition starting from a different reference edge would yield a tree with the same structure. Here, in order to simplify the description of the construction of SPQR-trees, we first describe them as rooted trees and then we comment on the implications of considering them as unrooted.

Rooted SPQR-Trees

The rooted SPQR-tree $\mathcal{T}_e$ of a biconnected graph G, with respect to a reference edge e, describes a recursive decomposition of G induced by its split pairs. The nodes of $\mathcal{T}_e$ are of four types: S, P, Q, and R. Their connections are called *arcs*, in order to distinguish them from the edges of G.

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

For each virtual edge e_i of $skel(\mu)$, recursively replace e_i with the skeleton $skel(\mu_i)$ of its corresponding child μ_i. The subgraph of G that is obtained in this way is the *pertinent graph* of μ and is denoted by $pertinent(\mu)$.

Given a biconnected graph G and a reference edge $e = (u', v')$, tree $\mathcal{T}_e$ is recursively defined as follows. At each step, a split component G^*, a pair of vertices $\{u, v\}$, and a node ν in $\mathcal{T}_e$ are given. A node μ corresponding to G^* is introduced in $\mathcal{T}_e$ and attached to its parent ν. Vertices u and v arethe *poles* of μ and 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 $G_1, \ldots, G_k$ $(k \geq 2)$ of G with respect to $\{u, v\}$, then μ is a P-node. Graph $skel(\mu)$ consists of k parallel virtual edges between u and v, denoted by

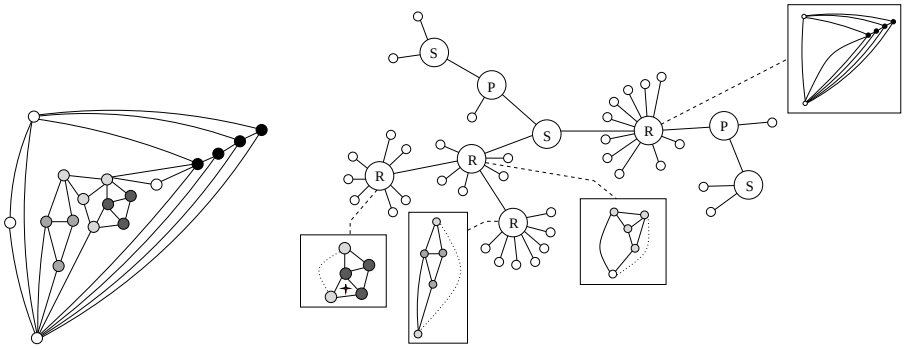
$e_1, \ldots, e_k$ and corresponding to $G_1, \ldots, G_k$, respectively. The decomposition recurs on $G_1, \ldots, G_k$, with $\{u, v\}$ as pair of vertices for every graph, and with μ as parent node.

Series Case: If G^* is composed of exactly one maximal split component of G with respect to $\{u, v\}$ and if G^* has cutvertices $c_1, \ldots, c_{k-1}$ ($k \geq 2$), appearing in this order on a path from u to v, then μ is an S-node. Graph *skel*(μ) is the path $e_1, \ldots, e_k$, where virtual edge e_i connects c_{i-1} with c_i ($i = 2, \ldots, k-1$), 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 $e_1, e_2, \ldots, e_{k-1}, e_k$ with μ as parent node, and with $\{u, c_1\}, \{c_1, c_2\}, \ldots, \{c_{k-2}, c_{k-1}\}, \{c_{k-1}, v\}$ 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 $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$ be the maximal split pairs of G^* ($k \geq 1$) and, for $i = 1, \ldots, k$, 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 *skel*(μ) 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.

For each node μ of $\mathcal{T}_e$, we add to *skel*(μ) the virtual edge (u, v) representing the parent of μ in $\mathcal{T}_e$. We say that an edge e' of G *projects* to a virtual edge e'' of *skel*(μ), for some node μ in $\mathcal{T}_e$, if e' belongs to the pertinent graph of the node of $\mathcal{T}_e$ corresponding to e''. Fig. 2.2 depicts a biconnected planar graph and its SPQR-tree.

Property 2.1 *Let C be a cycle of G and let μ be any node of $\mathcal{T}_e$. Then, either the edges of C belong to a single virtual edge of* skel(μ)*, or they belong to a set*



(a) (b)

Figure 2.2: (a) A biconnected planar graph and (b) its SPQR-tree, rooted at any Q-node adjacent to the R-node whose internal vertices are black. The skeletons of the internal R-nodes of the tree are represented inside the boxes. The virtual edge representing the parent of a node μ in the skeleton of μ is drawn as a dotted line.

of virtual edges that induce a cycle in skel(μ).

The SPQR-tree $\mathcal{T}_e$ of a graph G with n vertices and m edges has m Q-nodes and $O(n)$ S-, P-, and R-nodes. Also, the total number of vertices of the skeletons stored at the nodes of $\mathcal{T}_e$ is $O(n)$. Finally, SPQR-trees can be constructed and handled efficiently. Namely, given a biconnected planar graph G, the SPQR-tree $\mathcal{T}_e$ of G can be computed in linear time [GM00].

Unrooted SPQR-Trees

Now we describe how to modify $\mathcal{T}_e$ in order to have an unrooted tree $\mathcal{T}$.

Trees $\mathcal{T}_e$ and $\mathcal{T}$ have the same nodes and arcs. They only differ in the skeleton of their nodes. Given a node μ_e of $\mathcal{T}_e$, with split pair $\{u, v\}$ and parent ν_e, the virtual edge (u, v) is added to *skel*(μ_e) to obtain *skel*(μ) in $\mathcal{T}$.

Hence, in an unrooted SPQR-tree $\mathcal{T}$, an arc (μ, ν) identifies two virtual edges $e(\nu|\mu) \in$ *skel*(μ) and $e(\mu|\nu) \in$ *skel*(ν) that represent how the split component ν "attaches" to *skel*(μ) and how the split component μ "attaches" to *skel*(ν), respectively.

For unrooted SPQR-trees an equivalent concept of pertinent graph is defined. Namely, a *merge operation* of the skeletons of two adjacent nodes μ

and ν consists of removing $e(\nu|\mu)$ from $skel(\mu)$ and $e(\mu|\nu)$ from $skel(\nu)$ and identifying their corresponding end-vertices. Given a node μ of $\mathcal{T}$, the whole graph G can be obtained by recursively merging μ with its adjacent nodes. We call such an operation *merging of* $\mathcal{T}$. If $e(\nu|\mu)$ is a virtual edge of $skel(\mu)$, we define $pertinent(\mu, e(\nu|\mu))$ as the subgraph obtained by removing $e(\nu|\mu)$ and recursively merging μ with its adjacent nodes with the exception of ν. The graph $pertinent(\mu, e(\nu|\mu))$ defined on $\mathcal{T}$ coincides with the graph $pertinent(\mu)$ defined on $\mathcal{T}_e$ when ν is on the path from μ to the root of $\mathcal{T}_e$.

Observe that, two SPQR-trees $\mathcal{T}_{e'}$ and $\mathcal{T}_{e''}$, rooted at two different edges e' and e'' of G, correspond to the same unrooted tree $\mathcal{T}$.

Using SPQR-Trees to Represent Planar Embeddings

Let G be a biconnected graph and let $\mathcal{T}$ be the SPQR-tree of G. Graph G is planar if and only if the skeletons of all the nodes of $\mathcal{T}$ are planar [BDD00].

The SPQR-tree $\mathcal{T}$ can be used to represent all the planar embeddings of G. In fact, suppose that one of the combinatorial embeddings of the skeleton of each node is chosen. A combinatorial embedding of G can be obtained by merging the skeletons of all the adjacent nodes of $\mathcal{T}$ while preserving their embedding.

Observe that:

(i) the skeleton of an S-node admits exactly one combinatorial embedding (each vertex has degree two);

(ii) the skeleton of a P-node admits as many combinatorial embeddings as the number of permutations of its virtual edges; and

(iii) the skeleton of an R-node, which is triconnected, admits exactly one combinatorial embedding up to a reversal of the adjacency lists of its vertices.

Hence, a combinatorial embedding Γ of G is identified by specifying for each R-node one of its two possible embeddings and for each P-node a permutation of its adjacent nodes, as described in [BDD00]. We have the following.

Property 2.2 *A planar embedding of the skeleton of every node of* $\mathcal{T}$ *determines a planar embedding of* G *and vice versa.*

In order to represent the external face f in the SPQR-tree $\mathcal{T}$, observe that a face is uniquely identified by the set of edges incident to it, with the only exception, which can be easily handled, of the case when G is a simple cycle.

Given a face f of a combinatorial embedding Γ, the unique subtree of $\mathcal{T}$ whose leaves are the Q-nodes incident to f is called the *allocation tree* of f. Each node of the allocation tree of f is an *allocation node* of f. Figure 2.3 shows examples of allocation trees.

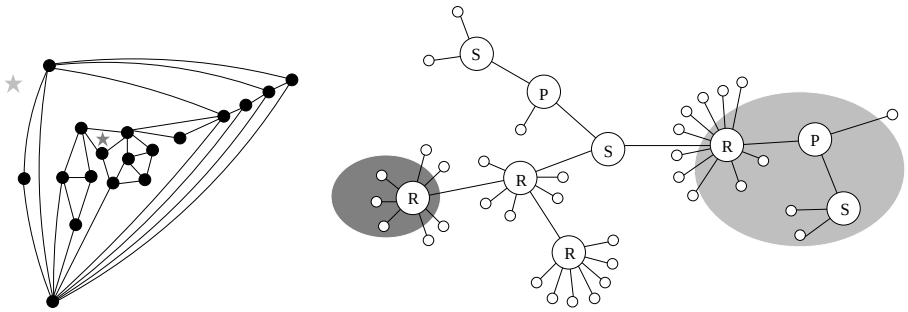


Figure 2.3: (a) A biconnected planar graph G and (b) the allocation trees of the two faces of G marked with a star.

Let μ be an allocation node of f. In *skel*(μ) there exists exactly one face f_μ that "corresponds to" f, in the sense that f_μ will be transformed into f when the merging of $\mathcal{T}$ is performed. We call f_μ the *representative* of f in *skel*(μ). In the following, we will denote by f both a face of Γ and its representative face in the skeleton of one of its allocation nodes. We say that f *belongs* to all its allocation nodes. Observe that, if the graph is a simple cycle, it contains only two faces, whose allocation tree is the whole SPQR-tree.

Therefore, a planar embedding $\langle \Gamma, f \rangle$ can be represented by a suitable labeling of $\mathcal{T}$. Namely, each R-node is labeled with a Boolean value and each P-node with the circular ordering of its adjacent nodes. The external face f is represented by its allocation tree.

The following lemma shows the relationship between faces belonging to nodes that are adjacent in $\mathcal{T}$.

Lemma 2.1 *Let μ and ν be two adjacent nodes of an SPQR-tree $\mathcal{T}$ of a graph G with planar embedding $\langle \Gamma, f \rangle$.*

1. *There are exactly two faces f' and f'' of Γ that belong both to μ and to ν.*
2. *In* skel(μ) *(*skel(ν)*) f' and f'' share edge $e(\nu|\mu)$ $(e(\mu|\nu))$.*

3. *If μ (ν) is not an S-node, then $e(\nu|\mu)$ ($e(\mu|\nu)$) is the only edge shared by f' and f'' in* skel(μ) (skel(ν)).

Proof: Observe that when merging the skeletons of μ and ν the two faces incident to $e(\nu|\mu)$ are identified with the two faces incident to $e(\mu|\nu)$, and such faces correspond to the same faces f' and f'' of Γ. Also, $skel(\mu)$ and $skel(\nu)$ do not share other faces.

Faces f' and f'' are represented in $skel(\mu)$ ($skel(\nu)$) by the two faces adjacent to $e(\nu|\mu)$ ($e(\mu|\nu)$). Only if a node, say μ, is an S-node, f' and f'' can also share other edges further than $e(\nu|\mu)$, that is, the virtual edges representing the nodes that are adjacent to μ in $\mathcal{T}$. □

Part II

Planar Embeddings

Chapter 3

Topological Morphing of Planar Graphs

In this chapter[1] we analyze the relationships among different planar embeddings of the same graph and study how two planar embeddings can be morphed one into the other with the minimum number of elementary changes, while preserving the mental map of the user. We call this problem *Topological Morphing*, in analogy with the well-known *Geometric Morphing* problem, in which it is studied how two planar drawings can be morphed one into the other with the minimum number of elementary changes.

First, we have to decide which elementary changes are admitted in order to preserve the mental map; then, we have to define which operations better describe such changes; finally, we have to study the problem of minimizing the number of these operations. We define two operations for morphing embeddings that describe natural transformations and we show that the problem of morphing embeddings with the minimum number of such operations is NP-hard for biconnected planar graphs. Further, we give polynomial-time algorithms for some restricted versions of the problem and, based on such algorithms, we give a fixed parameter tractable algorithm for the general case.

[1] Part of the work presented in this chapter is a joint work with Pier Francesco Cortese, Giuseppe Di Battista, and Maurizio Patrignani, appeared in [ACBP08].

3.1 Introduction

A useful feature of a graph drawing editor is the possibility of selecting a certain face of the drawing and promoting it to be the external face (for instance, see [dO]). In order to preserve the mental map, the user would like the editor to execute such an operation with as few changes to the drawing as possible.

The above operation is just an example of a topological feature that would be useful to have at disposal from an editor. More generally, it would be interesting to have an editor allowing the user to look at a drawing and to specify in some way, e.g. pointing at vertices or edges, a new embedding. Such an embedding could be even requested at a more abstract level, asking the editor to go to one with minimum depth, or with minimum radius, etc. Again, the editor should transform the current embedding into the new one smoothly, i.e., with the minimum number of changes.

A similar problem occurs when an editor has to geometrically morph a drawing into another one, specified in some way by the user, while keeping the topology unchanged. In this case, the operations performed by the editor are topology-preserving translations and scaling of objects. Again, the user would like to see the minimum number of intermediate snapshots.

The existence of a geometric morphing between two drawings was addressed surprisingly long ago. In 1944, Cairns proved that between any two straight-line drawings of a triangulation there exists a morph in which any intermediate drawing is straight-line planar [Cai44]. This result was extended to general planar graphs by Thomassen in 1983 [Tho83]. The first algorithms to find such morphings were proposed by Floater and Gotsman, in 1999, for triangulations [FG99] and by Gotsman and Surazhsky, in 2001, for general plane graphs [GS01]. While the search for a geometric straight-line morph between two given drawings of a planar graph with a polynomial number of steps and with a bounded size of the needed grid is still open, some recent studies address the problem for the special cases of poly-line morphing [LP08], orthogonal drawings [LPS06, BLS05], and arbitrary plane drawings [EKP03]. See [Lub07] for a complete survey on this subject.

We study the morphing between two drawings from the topological perspective and we call it *topological morphing*. There are many ways to state the problem, ranging from the family of graphs, to the admitted operations, to their completeness, to their ability to capture changes that are "natural" for the user, and to the metrics that distinguish a good morphing from a bad one.

In this work we make the following basic choices.

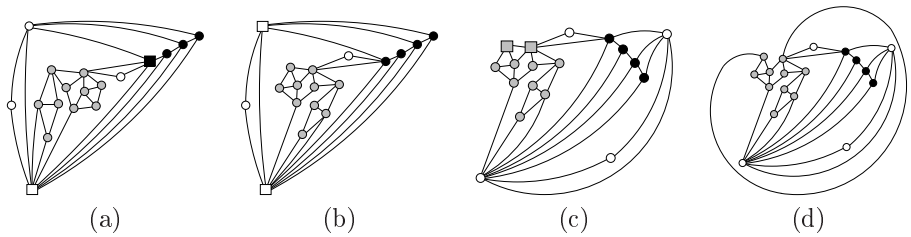


Figure 3.1: A sequence of flips and skips transforming a planar embedding.

1. We consider biconnected planar graphs, since this class of graphs is the building block of several graph drawing methodologies.

2. We consider operations that move entire blocks of the drawing, identified by some connectivity features, in one single step. Namely, using a term that is common in planarity testing literature, we call *flip* the operation that "flips" a component around its split pair. Also, borrowing the term from the common rope skipping game played by children, we call *skip* the operation that moves the external face to a selected face by "skipping" an entire component without modifying the combinatorial embedding. More formal definitions for the two operations will be given later.

3. We use the number of performed operations as the metric to evaluate morphings and we consider a topological morphing "good" if the editor can perform it with few flips and skips. Intuitively, the fewer operations are performed, the better the mental map of the user is preserved.

As an example, suppose that the graph is embedded as shown in Fig. 3.1(a) and that the user would like to obtain the embedding in Fig. 3.1(d). A minimum sequence of operations that performs such a morphing consists of flipping the component separated by the square vertices of Fig. 3.1(a), then by skipping the component separated by the square vertices of Fig. 3.1(b), and finally by skipping the edge separated by the square vertices of Fig. 3.1(c).

We present the following results. Let G be a biconnected planar graph and denote by $\langle \Gamma, f \rangle$ one of its combinatorial embeddings Γ with f as external face. Suppose that $\langle \Gamma_1, f_1 \rangle$ is the current topology and that $\langle \Gamma_2, f_2 \rangle$ represents a target topology chosen by the user. In Sect. 3.3 we show that, if both flips and skips are allowed, the general problem of morphing $\langle \Gamma_1, f_1 \rangle$ into $\langle \Gamma_2, f_2 \rangle$

with the minimum number of flips and skips is NP-complete. Motivated by such a result we tackle several restricted problems. In Sect. 3.4 we give a linear time algorithm to move the external face from f_1 to f_2 with the minimum number of operation when $\Gamma_1 = \Gamma_2$ and only skips are allowed. In Sect. 3.5 we show that the topological morphing problem can be efficiently solved if G does not contain any parallel triconnected components. In Sect. 3.6 we show that the problem is fixed-parameter tractable. Definitions and properties of basic operations are in Sect. 3.2, while concluding remarks are in Sect. 3.7.

3.2 Flip and Skip Operations

Let G be a biconnected planar graph and let $\langle \Gamma, f \rangle$ be a planar embedding of G, where Γ is a combinatorial embedding and f is the external face.

In the following we formally define the flip and skip operations, that can be used to modify $\langle \Gamma, f \rangle$ and, as a consequence, the labeling of the SPQR-tree $\mathcal{T}$ of G representing $\langle \Gamma, f \rangle$. Intuitively, a flip operation "flips" a component around its split pair, while a skip operation allows the external face to "skip" an entire component, promoting a new external face without modifying the combinatorial embedding.

First, we define the flip operation with respect to a planar embedding $\langle \Gamma, f \rangle$. Second, we show how it modifies the labeling of $\mathcal{T}$ representing $\langle \Gamma, f \rangle$.

Let $\{u, v\}$ be a maximal split pair of G and let G_1^i, with $i = 1 \dots q$, be a set of topologically contiguous maximal split components of G w.r.t. $\{u, v\}$. Let $G_1 = \bigcup_1^q G_1^i$ be the subgraph of G obtained as the union of all the G_1^i. We define the *flip* operation on $\langle \Gamma, f \rangle$ with respect to G_1: $\mathit{flip}(\langle \Gamma, f \rangle, G_1) = \langle \Gamma', f' \rangle$, where Γ' is obtained from Γ by reversing the adjacency lists of all the vertices of G_1, but for u and v, and by reversing the order of the edges of G_1 in the adjacency lists of u and v. Face f' is determined as follows. If at least one out of u and v is not in f, then $f' = f$. Otherwise, f' is the unique face of Γ' containing both the edges belonging to f and not belonging to G_1, and some edges of G_1 not belonging to f.

As an example, see the flip operation applied to the embedding of Fig. 3.1(a) that yields the embedding of Fig. 3.1(b).

Now we show how the labeling of $\mathcal{T}$ is modified by a flip operation $\mathit{flip}(\langle \Gamma, f \rangle, G_1)$. For each maximal split component G_1^i of G, with $1 \leq i \leq q$, with respect to $\{u, v\}$ contained into G_1, consider the node μ_i of $\mathcal{T}$ such that $\mathit{pertinent}(\mu_i, e(\nu_i|\mu_i)) = G_1^i$, where ν_i is a node adjacent to μ_i in $\mathcal{T}$. Observe that $q > 1$ implies that ν_i is the same P-node μ_P for each component G_1^i, whereas

$q = 1$ implies that $\nu_s \neq \nu_t$ for each $1 \leq s, t \leq q$ and $s \neq t$. Further, consider the subtree $\mathcal{T}_1^i$ of $\mathcal{T}$ rooted at μ_i and not containing ν_i. Then, when operation $\mathit{flip}(\langle \Gamma, f \rangle, G_1) = \langle \Gamma', f' \rangle$ is performed, the labeling of $\mathcal{T}$ representing Γ' is obtained from the labeling of $\mathcal{T}$ representing Γ by complementing the Boolean value of all the nodes of $\mathcal{T}_1^i$, for each $i = 1 \dots q$, and, if $q > 1$, by reversing the subsequence corresponding to $\mu_1, \mu_2, \dots, \mu_q$ in the circular ordering of the adjacent nodes of μ_P. If G_1 contains some edges belonging to f, then the new external face f' is obtained as previously described and the new allocation tree must be evaluated according to it.

With the intent of maintaining the mental map of the user, we add the constraint that a flip operation $\mathit{flip}(\langle \Gamma, f \rangle, G_1)$ cannot be performed if G_1 contains all the edges of the external face f, because we regard that a flipping of the entire external structure of the graph around an internal component is undesirable from a comprehension point of view.

The following properties describe three basic features of the flip operation.

Property 3.1 $\text{flip}(\mathit{flip}(\langle \Gamma, f \rangle, G_1), G_1) = \langle \Gamma, f \rangle$.

Property 3.2 *If G_1 is a path, then* $\text{flip}(\langle \Gamma, f \rangle, G_1) = \langle \Gamma, f \rangle$.

Property 3.3 *If $G - G_1$ is a path, then* $\text{flip}(\langle \Gamma, f \rangle, G_1) = \langle \overline{\Gamma}, f \rangle$, *where $\overline{\Gamma}$ is obtained from Γ by reversing the adjacency lists of all the vertices.*

Let Γ_1 be a combinatorial embedding of G and let Γ_2 be a "target" combinatorial embedding of G. It is easy to see that it is always possible to find a sequence of flip operations that leads from $\langle \Gamma_1, f_1 \rangle$, for an arbitrary $f_1 \in \Gamma_1$, to $\langle \Gamma_2, f_2 \rangle$, for a suitable $f_2 \in \Gamma_2$. Also, such a sequence is composed of a linear number of operations and can be computed in linear time. In fact, flip operations allow both to arbitrarily permute the circular list associated with each P-node and to reverse the adjacency lists of the skeleton of each R-node. We formalize this concept in the following lemma.

Denote by $\mathcal{F}(\Gamma_1, \Gamma_2)$ the minimum number of flips to obtain $\langle \Gamma_2, f_2 \rangle$ from $\langle \Gamma_1, f_1 \rangle$, for an arbitrary $f_1 \in \Gamma_1$ and a suitable $f_2 \in \Gamma_2$.

Lemma 3.1 *For any two combinatorial embeddings Γ_1 and Γ_2 of G we have that $\mathcal{F}(\Gamma_1, \Gamma_2)$ is $O(n)$, where n is the number of vertices of G.*

Proof: Consider the labeling of the SPQR-tree of G representing Γ_1. It is easy to find a sequence of flips such that each flip either gives to an R-node the label that it has in Γ_2 or places the neighbor of a P-node μ in its position in the

circular order of μ in Γ_2. The statement follows from the fact that the number of nodes in the SPQR-Tree is $O(n)$ and that each operation can be performed in constant time. □

Now we define the skip operation, which allows to modify the external face of a planar embedding. As for the flip operation, we first define the skip operation with respect to a planar embedding $\langle \Gamma, f \rangle$ and then we show how such an operation modifies the labeling of $\mathcal{T}$ representing $\langle \Gamma, f \rangle$.

Let G be a planar graph and let $\langle \Gamma, f_1 \rangle$ be one of its planar embeddings. Let $\{u, v\}$ be a split pair of G incident to f_1 and to another face $f_2 \in \Gamma$. Skip is defined as follows: $\mathit{skip}(\langle \Gamma, f_1 \rangle, f_2) = \langle \Gamma, f_2 \rangle$.

As an example, see the skip operation applied to the embedding of Fig. 3.1(b) that yields the embedding of Fig. 3.1(c).

When a skip operation is applied to a planar embedding $\langle \Gamma, f \rangle$ and a face f' sharing a split pair $\{u, v\}$ with f, Γ is not modified. Hence, a skip operation only acts on the labeling of the SPQR-tree $\mathcal{T}$ representing the current planar embedding by turning the current allocation tree into the one of f'.

Also, observe that a skip operation $\mathit{skip}(\langle \Gamma, f \rangle, f')$ corresponds to the skip of the virtual edge adjacent to both f and f' in one of the skeletons that contain both f and f' (see Lemma 2.1). Such an observation is the main reason why we permit the skip of an entire component of the graph with a single operation.

Let $\langle \Gamma, f_1 \rangle$ be a planar embedding of G and let f_2 be any "target" internal face of Γ. It is easy to see that it is always possible to find a sequence of skip operations leading from $\langle \Gamma, f_1 \rangle$ to $\langle \Gamma, f_2 \rangle$. Also, such a sequence is composed of a linear number of operations and can be computed in linear time. In fact, a sequence of skips leading from f_1 to f_2 can be obtained as a path on the dual of G connecting f_1 and f_2. We formalize this concept in the following lemma.

Denote by $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$ the minimum number of skips to obtain $\langle \Gamma, f_2 \rangle$ from $\langle \Gamma, f_1 \rangle$.

Lemma 3.2 *Let $\langle \Gamma, f_1 \rangle$ be a planar embedding of G. For any face $f_2 \in \Gamma$, we have that $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$ is $O(n)$, where n is the number of vertices of G.*

Proof: Consider the dual graph D of G when the combinatorial embedding is Γ. A path in D from the vertex representing f_1 to the vertex representing f_2 corresponds to a sequence of skips which moves the external face from f_1 to f_2. Hence, the number of edges of G, that is $O(n)$, is an upper bound on the value of $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$. □

In Sect.3.4 we will show that, given a planar embedding $\langle \Gamma, f_1 \rangle$ of a biconnected planar graph G and an internal face $f_2 \in \Gamma$, $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$ can be

computed in linear time.

Given two planar embeddings $\langle \Gamma_1, f_1 \rangle$ and $\langle \Gamma_2, f_2 \rangle$ of a graph G, one could ask which is the minimum number of flip and skip operations for obtaining $\langle \Gamma_2, f_2 \rangle$ from $\langle \Gamma_1, f_1 \rangle$. We denote such a number by $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$.

A trivial upper bound to $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ can be found by first performing all the flips needed to transform Γ_1 into Γ_2, hence obtaining a new external face f_1' which possibly coincides with f_1, and then by applying an optimal sequence of skips to transform the resulting embedding $\langle \Gamma_2, f_1' \rangle$ into $\langle \Gamma_2, f_2 \rangle$. However, in Sect. 3.5 we will show that the obtained bound can be far from the optimum (see Fig. 3.5). Property 3.4 formalizes this upper bound.

Property 3.4 *$\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle) \leq \mathcal{F}(\Gamma_1, \Gamma_2) + \mathcal{S}(\langle \Gamma_2, f_1' \rangle, \langle \Gamma_2, f_2 \rangle)$, where f_1' is the external face obtained when computing $\mathcal{F}(\Gamma_1, \Gamma_2)$.*

Lemma 3.3 *For any two combinatorial embeddings Γ_1, Γ_2 and any two faces $f_1 \in \Gamma_1$ and $f_2 \in \Gamma_2$, we have that $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ is $O(n)$, where n is the number of vertices of G.*

Proof: By Property 3.4, $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle) \leq \mathcal{F}(\Gamma_1, \Gamma_2) + \mathcal{S}(\langle \Gamma_2, f_1' \rangle, \langle \Gamma_2, f_2 \rangle)$. The statement follows from the fact that both $\mathcal{F}(\Gamma_1, \Gamma_2)$ and $\mathcal{S}(\langle \Gamma_2, f_1' \rangle, \langle \Gamma_2, f_2 \rangle)$ are $O(n)$, by Lemmata 3.1 and 3.2, respectively. □

3.3 NP-Completeness of the General Case

In this section we prove that, given a biconnected planar graph G and two of its planar embeddings $\langle \Gamma_1, f_1 \rangle$ and $\langle \Gamma_2, f_2 \rangle$, the problem of transforming $\langle \Gamma_1, f_1 \rangle$ into $\langle \Gamma_2, f_2 \rangle$ with the minimum number of flip and skip operations is NP-complete.

Lemma 3.4 *Let G be a biconnected planar graph and let $\langle \Gamma_1, f_1 \rangle$ and $\langle \Gamma_2, f_2 \rangle$ be two planar embeddings of G. The problem of computing $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ belongs to class NP.*

Proof: By Lemma 3.3, the transformation of $\langle \Gamma_1, f_1 \rangle$ into $\langle \Gamma_2, f_2 \rangle$ requires at most $O(n)$ steps, where n is the number of vertices of G. At each step, either a flip or a skip is performed. The number of split pairs is $O(n^2)$, and each split pair separates $O(n)$ maximal split components. Hence, for each split pair, the number of consecutive maximal split components that can be

flipped/skipped is $O(n^2)$. Therefore, a polynomial number of skip/flip operations can be performed at each of the $O(n)$ steps that transform $\langle \Gamma_1, f_1 \rangle$ into $\langle \Gamma_2, f_2 \rangle$. A non-deterministic Turing machine could non-deterministically perform all the possible operations and check, for each of the obtained embeddings, if it coincides with $\langle \Gamma_2, f_2 \rangle$. □

In order to prove that the problem of computing $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ is NP-hard, we introduce a problem, called *Sorting by Reversals* (SBR), which has been deeply studied by the Computational Biology community.

An instance of SBR is a linear permutation $\sigma = i_1, i_2, \ldots, i_n$ of the first n positive integers $1, 2, \ldots, n$. A *reversal* operation on σ consists of replacing a subsequence $i_k, i_{k+1}, \ldots, i_{k+h}$ with the reversed subsequence $i_{k+h}, i_{k+h-1}, \ldots, i_k$. The goal of the problem is to obtain the sorted sequence $1, 2, \ldots, n$ from σ with the minimum number of reversal operations. Problem SBR can be also defined on circular permutations, where the reversal operation can be applied on circular subsequences, too. In [MWD00] and [SSL03] it has been shown that the two problems are equivalent, while in [Cap97] Caprara showed that both the problems are NP-complete.

Theorem 3.1 *Let G be a biconnected planar graph and let $\langle \Gamma_1, f_1 \rangle$ and $\langle \Gamma_2, f_2 \rangle$ be two planar embeddings of G. The problem of computing $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ is NP-complete.*

Proof: The proof is obtained by reducing SBR to the problem of computing $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$. Given an instance σ of SBR, we construct a biconnected planar graph G and two embeddings $\langle \Gamma_1, f_1 \rangle$ and $\langle \Gamma_2, f_2 \rangle$ such that $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ is the minimum number of reversals that order σ.

Let $\sigma = i_1, i_2, \ldots, i_n$ be an instance of SBR. In order to construct G, consider a triconnected embedded graph with one face h at distance n from the external face f. Observe that $O(n)$ vertices are sufficient to build such a graph. Consider two arbitrary vertices u and v incident to h. For each $j = 1, 2, \ldots, n$, create a path p_j with j internal vertices connecting u and v. The order of such paths around u is $p_{i_1}, p_{i_2}, \ldots, p_{i_n}$ in Γ_1 and $p_1, p_2, \ldots, p_n$ in Γ_2. An example is given in Fig. 3.2, where the graph G corresponding to sequence $\sigma = 1, 3, 4, 2$ is shown.

We now prove that the sequence of $\mathcal{FS}(\langle \Gamma_1, f \rangle, \langle \Gamma_2, f \rangle)$ operations needed to transform $\langle \Gamma_1, f \rangle$ into $\langle \Gamma_2, f \rangle$ can be mapped into the minimum sequence of reversals that orders σ, and vice versa. First, observe that $\mathcal{FS}(\langle \Gamma_1, f \rangle, \langle \Gamma_2, f \rangle) \leq n$. In fact, the paths can be ordered with at most n flips, where the j-th flip places p_j at position j. Further, observe that, since face h is at dis-

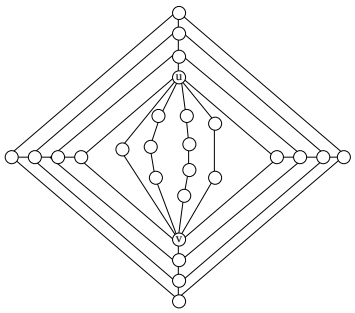


Figure 3.2: The graph G corresponding to sequence $\sigma = 1, 3, 4, 2$.

tance n from the external face, performing some skip operations to simplify the ordering of the paths $p_1, \ldots p_n$ would imply a sequence of at least n skips to reach h and a sequence of at least n skips to go back to f. Hence, any minimum sequence of flip and skip operations that transforms $\langle \Gamma_1, f \rangle$ into $\langle \Gamma_2, f \rangle$ does not contain any skip operations. Finally, observe that a flip operation on paths $p_{i_k}, p_{i_{k+1}}, \ldots, p_{i_{k+h}}$ corresponds to a reversal of a subsequence $i_k, i_{k+1}, \ldots, i_{k+h}$ and that, since each path p_i corresponds to an S-node with $i + 1$ adjacent Q-nodes, the labeling of the SPQR-tree is unchanged, except for the ordering of the adjacent nodes of the P-node induced by the split pair $\{u, v\}$. In order to prove that the construction of G, $\langle \Gamma_1, f \rangle$ and $\langle \Gamma_2, f \rangle$ can be performed in polynomial time, we observe that the triconnected embedded graph with one face h at distance n from the external face contains $O(n)$ vertices and that the total number of vertices of paths p_j, with $j = 1, 2, \ldots, n$, is $\sum_{i=1}^{n} i = n(n-1)/2$. Hence, the problem of computing $\mathcal{FS}(\langle \Gamma_1, f \rangle, \langle \Gamma_2, f \rangle)$ is NP-hard. Since Lemma 3.4 guarantees that the problem belongs to class NP, the statement follows. □

Since the reduction from SBR to the problem of determining $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ only relies on the correspondence between a reversal and a flip and that skips are never performed, it is possible to use the same technique to reduce SBR to the problem of determining $\mathcal{F}(\Gamma_1, \Gamma_2)$. In this case, we do not need to place vertices u and v on a face that is at distance n from the external face, since skip operations are not allowed.

3.4 Linearity of the Case with Fixed Combinatorial Embedding

Let G be a biconnected planar graph, and let $\langle \Gamma, f_1 \rangle$ and $\langle \Gamma, f_2 \rangle$ be two planar embeddings of G. In this section, we show how to compute the value of $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$. Fig. 3.3 shows an example of a shortest sequence of skips moving the external face of an embedded graph to a selected one.

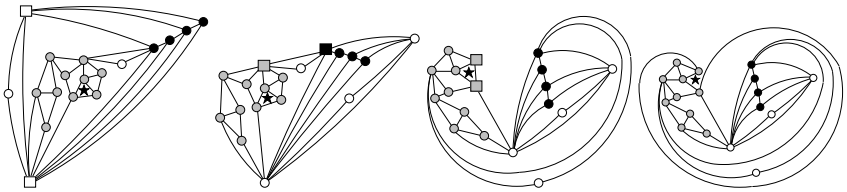


Figure 3.3: A sequence of three skip operations moving the external face to the face marked by a star. Square vertices represent the split pairs identifying the components that are skipped at each of the steps.

First, we need to introduce the following lemma.

Lemma 3.5 *Let G be a biconnected planar graph and let $\mathcal{T}$ be the SPQR-tree of G. Let $\langle \Gamma, f_1 \rangle$ and $\langle \Gamma, f_2 \rangle$ be two planar embeddings of G. If there exists an R-node μ of $\mathcal{T}$ such that* skel(μ) *contains both f_1 and f_2, then $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$ is the length of the shortest path from f_1 to f_2 on the dual of* skel(μ).

Proof: First we remind that, since each pair of adjacent vertices in *skel*(μ) corresponds to a split pair of G, the external face can be moved to an adjacent face of the skeleton with a single skip operation. Hence, the length of a shortest path on the dual graph of *skel*(μ) from f_1 to f_2 is an upper bound for $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$. In order to show that such a length is also a lower bound, it suffices to observe that any path on the dual graph of Γ leading from f_1 to f_2 has to traverse the faces of *skel*(μ) from f_1 to f_2. □

Let $\mathcal{T}$ be the SPQR-tree of G and let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the allocation trees of f_1 and f_2, respectively. Based on the previous Lemma, we state that the value of $\mathcal{S} = \mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$ can be easily computed when $\mathcal{T}_1 \cap \mathcal{T}_2 \neq \emptyset$. Three subcases have to be distinguished: $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu\}$, $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu, \nu\}$, and

$\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu_1, \mu_2, \ldots, \mu_k\}$. Conversely, the case $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$ is more complex and will be analyzed in detail.

Case $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu\}$. In this case μ is the only node of $\mathcal{T}$ whose skeleton contains both f_1 and f_2. Observe that μ can not be an S-node, since otherwise all the nodes of $\mathcal{T}$ adjacent to μ would be in $\mathcal{T}_1 \cap \mathcal{T}_2$. If μ is a P-node, since a skip operation can move the external face from f_1 to any face of $skel(\mu)$, we have that $\mathcal{S} = 1$. If μ is an R-node, by Lemma 3.5, $\mathcal{S}$ is the length of the shortest path on the dual of $skel(\mu)$ from f_1 to f_2.

Case $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu, \nu\}$. In this case μ and ν are the only nodes of $\mathcal{T}$ whose skeleton contains both f_1 and f_2. Hence, they are adjacent in $\mathcal{T}$ and they can not be both P-nodes. Then, by Lemma 2.1, f_1 and f_2 are adjacent both in $skel(\mu)$ and in $skel(\nu)$ and therefore we have that $\mathcal{S} = 1$. Observe that, as in the previous case, neither μ nor ν can be S-nodes.

Case $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu_1, \mu_2, \ldots, \mu_k\}$**, with** $k \geq 3$. As this case is more involved, we treat it separately in the following lemma.

Lemma 3.6 *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the allocation trees of two faces f_1 and f_2 of a plane graph G, respectively. If $\mathcal{T}_1 \cap \mathcal{T}_2 = \mathcal{T}_3$, where $\mathcal{T}_3 = \{\mu_1, \mu_2, \ldots, \mu_k\}$ and $k \geq 3$, then $\mathcal{T}_3$ is a star graph whose central node is an S-node.*

Proof: First, we show that the second node in each path of length 3 in $\mathcal{T}_3$ is an S-node. Consider a path (μ_a, μ_b, μ_c) in $\mathcal{T}_3$ and suppose, for a contradiction, that μ_b is not an S-node. By Lemma 2.1, the two adjacent nodes μ_a and μ_b share exactly two faces $f'_{a,b}$ and $f''_{a,b}$ and the two adjacent nodes μ_b and μ_c share exactly two faces $f'_{b,c}$ and $f''_{b,c}$. Also, by Lemma 2.1, since μ_b is not an S-node, in $skel(\mu_b)$ $f'_{a,b}$ and $f''_{a,b}$ share only virtual edge $e(\mu_a|\mu_b)$, while $f'_{b,c}$ and $f''_{b,c}$ share only virtual edge $e(\mu_c|\mu_b)$. Since $e(\mu_a|\mu_b) \neq e(\mu_c|\mu_b)$, we have that $\{f'_{a,b}, f''_{a,b}\} \neq \{f'_{b,c}, f''_{b,c}\}$; hence $|\{f'_{a,b}, f''_{a,b}\} \cap \{f'_{b,c}, f''_{b,c}\}| \leq 1$, contradicting the fact that μ_a, μ_b, and μ_c share the two faces f_1 and f_2. Hence, we conclude that the second node of each path of length three in $\mathcal{T}_3$ is an S-node.

Second, we show that $\mathcal{T}_3$ contains exactly one S-node. Since $\mathcal{T}_3$ has at least three nodes, it contains at least a path of length 3, whose second node is an S-node. Suppose, by contradiction, that $\mathcal{T}_3$ contains two S-nodes ν_1 and ν_2. Since all the nodes of $\mathcal{T}_3$ share the two faces f_1 and f_2 and each S-node has exactly two faces, the two S-nodes ν_1 and ν_2 are adjacent in the SPQR-tree of G, a contradiction. Hence, $\mathcal{T}_3$ contains exactly one S-node ν_S and, since in each path of length 3 the second node is an S-node, each path in $\mathcal{T}_3$ has length at most 3; hence, $\mathcal{T}_3$ is a star graph with ν_S as central node. □

By Lemma 3.6 and by the fact that f_1 and f_2 are faces of the skeleton of the same S-node, it follows that $\mathcal{S} = 1$.

In the case $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$, the computation of $\mathcal{S}$ is not trivial; however, we provide a linear time algorithm, called SkipOnly, to solve this problem. The algorithm is described below and its pseudo-code is provided in Algorithm 1.

Algorithm 1 SkipOnly

Require: A biconnected planar graph G and two of its planar embeddings $\langle \Gamma, f_1 \rangle$ and $\langle \Gamma, f_2 \rangle$.

Ensure: The minimum length sequence of $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$ skip operations to obtain $\langle \Gamma, f_2 \rangle$ from $\langle \Gamma, f_1 \rangle$.

Prepocessing Phase

1: Compute the SPQR-tree $\mathcal{T}$ of G

2: Compute the allocation trees $\mathcal{T}_1$ and $\mathcal{T}_2$ of f_1 and f_2

Computation Phase

3: Compute the skip path $sp(f_1, f_2)$ between $\mathcal{T}_1$ and $\mathcal{T}_2$

4: Construct the track graph $Track(f_1, f_2)$

5: Compute the weighted shortest path on $Track(f_1, f_2)$ from f_1 to f_2

Execution Phase

6: Perform the sequence of skip operations on G that determined the weight of the edges of the weighted shortest path. An horizontal edge corresponds to a skip of a whole component, while a vertical edge corresponds to a shortest path on the dual of the skeleton of a component.

Suppose $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$. We call *skip path* the (unique) shortest path in $\mathcal{T}$ between any node of $\mathcal{T}_1$ and any node of $\mathcal{T}_2$ (see Fig. 3.4(a)). We denote such a skip path by $sp(f_1, f_2)$.

Since a skip operation from a face of $skel(\mu)$ to a face of $skel(\nu)$ is admitted only if μ and ν are adjacent in $\mathcal{T}$, the following property holds.

Property 3.5 *Any sequence of skip operations moving the external face from f_1 to f_2 traverses all the nodes of the skip path* $\mathrm{sp}(f_1, f_2)$ *between $\mathcal{T}_1$ and $\mathcal{T}_2$.*

In order to compute the sequence of $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$ skips moving the external face from f_1 to f_2, we define a *weighted track graph* [BBF04] $Track(f_1, f_2)$ (see Fig. 3.4(b)). Nodes of $Track(f_1, f_2)$ correspond to faces of the skeletons of the nodes in $sp(f_1, f_2)$. In particular, let $sp(f_1, f_2) = \{\mu_1, \ldots \mu_k\}$ be the

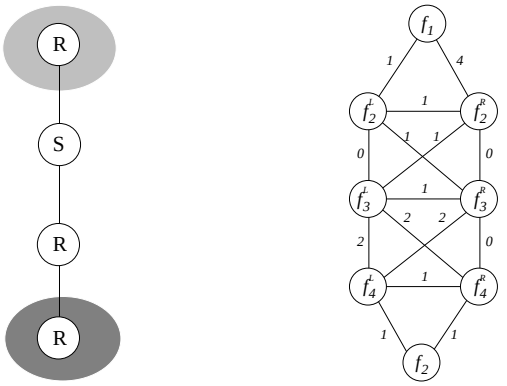


Figure 3.4: (a) The skip path connecting the allocation trees of Fig. 2.3 and (b) the corresponding weighted track graph.

skip path connecting a node μ_1 of the allocation tree of f_1 to a node μ_k of the allocation tree of f_2. Observe that f_1 is the external face of $skel(\mu_1)$, while f_2 is an internal face of $skel(\mu_k)$. Track graph $Track(f_1, f_2)$ contains two nodes corresponding to f_1 and f_2. Also, for each node $\mu_i \in sp(f_1, f_2)$, $i = 2, \ldots, k$, $Track(f_1, f_2)$ contains two nodes ${f_i}^l$ and ${f_i}^r$ corresponding to the two faces of $skel(\mu_i)$ adjacent to the virtual edge $e(\mu_{i-1}|\mu_i)$ representing μ_{i-1} in $skel(\mu_i)$. Note that such faces correspond to the two faces of $skel(\mu_{i-1})$ adjacent to the virtual edge $e(\mu_i|\mu_{i-1})$ representing μ_i in $skel(\mu_{i-1})$.

Nodes of $Track(f_1, f_2)$ are assigned a leveling. The node corresponding to f_1 belongs to level 1, nodes ${f_i}^l$ and ${f_i}^r$, for $i = 2, \ldots, k$, belong to level i, and the node corresponding to f_2 belongs to level $k + 1$.

Edges of $Track(f_1, f_2)$ are of two types, *horizontal edges* and *vertical edges*. Horizontal edges connect nodes of the same level, while vertical edges connect nodes of adjacent levels. More precisely, horizontal edges are $({f_i}^r, {f_i}^l)$, for $i = 2, \ldots, k$, while vertical edges are $(f_1, {f_2}^l)$, $(f_1, {f_2}^r)$, $({f_k}^l, f_2)$, $({f_k}^r, f_2)$ and, for $i = 2, \ldots, k-1$, $({f_i}^l, {f_{i+1}}^l)$, $({f_i}^l, {f_{i+1}}^r)$, $({f_i}^r, {f_{i+1}}^l)$, and $({f_i}^r, {f_{i+1}}^r)$.

Edges of $Track(f_1, f_2)$ are weighted. The weight of an edge (f', f'') in $Track(f_1, f_2)$ represents the number of skip operations needed to move the external face from f' to f''. Horizontal edges $({f_i}^r, {f_i}^l)$ are weighted 1. Such a weight represents the possibility to skip virtual edge $e(\mu_{i-1}|\mu_i)$ in $skel(\mu_i)$ to

move from ${f_i}^r$ to ${f_i}^l$ (or from ${f_i}^l$ to ${f_i}^r$).

In order to assign a weight to the vertical edges, consider an edge $(f_i^{y_i}, f_{i+1}^{y_{i+1}})$, with $y_i, y_{i+1} \in \{l, r\}$, spanning levels i and $i+1$. If μ_i is a P-node, then the weight is 0 if μ_{i-1} and μ_{i+1} are consecutive in the circular ordering of the nodes adjacent to μ_i and $y_i \neq y_{i+1}$, and it is 1 otherwise, since all the faces of the skeleton of a P-node are adjacent to the same split pair. If μ_i is an S-node, then the weight is 0 if $y_i = y_{i+1}$ and is 1 otherwise, since only two faces exist in the skeleton of an S-node. Finally, if μ_i is an R-node, then the weight of $(f_i^{y_i}, f_{i+1}^{y_{i+1}})$ is the length of the shortest path from $f_i^{y_i}$ to $f_{i+1}^{y_{i+1}}$ on the dual of $skel(\mu_i)$, where edge $({f_i}^r, {f_i}^l)$ has been removed, because the possibility of skipping such an edge is already taken into account by the horizontal edge $({f_i}^r, {f_i}^l)$ of $Track(f_1, f_2)$.

We compute a weighted shortest path from the node corresponding to f_1 to the node corresponding to f_2 on $Track(f_1, f_2)$. The nodes of $Track(f_1, f_2)$ that are traversed during such a shortest path are the faces of G that have to be traversed when morphing $\langle \Gamma, f_1 \rangle$ into $\langle \Gamma, f_2 \rangle$ with the minimum number $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$ of steps. The sequence of skips to be performed is given by the operations that determined the weight of the edges of $Track(f_1, f_2)$ that are traversed by the weighted shortest path.

Theorem 3.2 *Let G be a biconnected planar graph, and let $\langle \Gamma, f_1 \rangle$ and $\langle \Gamma, f_2 \rangle$ be two planar embeddings of G. If only skip operations are allowed, then there exists an algorithm to compute $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$ in linear time.*

Proof: Apply Algorithm SkipOnly. Consider the weighted shortest path on $Track(f_1, f_2)$ from f_1 to f_2. Such a path, by construction, corresponds to a sequence s of skip operations that moves the external face of G from f_1 to f_2. Suppose, for a contradiction, that there exists a sequence s^* of skips from f_1 to f_2 on G such that $|s^*| < |s|$. Since, by Property 3.5, all the nodes of $sp(f_1, f_2)$ have to be traversed when moving the external face from f_1 to f_2, s^* traverses at least one face ${f_i}^y$, with $y \in \{l, r\}$, for each node $\mu_i \in sp(f_1, f_2)$. Hence, we have that such faces partition s^* into $|sp(f_1, f_2)|$ subsequences $s_i^* = \{f_i^y, x_i^1, x_i^2, \ldots, x_i^j, f_{i+1}^y\}$, where x_i^1 may possibly be the other face in $skel(\mu_i)$ adjacent to the virtual edge representing μ_{i+1}, when the horizontal edge $({f_i}^r, {f_i}^l)$ is traversed. Since, by hypothesis, $|s^*| < |s|$, there exists a subsequence $s_i^* = \{f_i^y, x_i^1, x_i^2, \ldots, x_i^j, f_{i+1}^y\}$ such that $|s_i^*| < w(f_i^y, f_{i+1}^y)$, where $w(f_i^y, f_{i+1}^y)$ is the weight of vertical edge (f_i^y, f_{i+1}^y).

If μ_i is a P-node or an S-node, then $w(f_i^y, f_{i+1}^y)$ is either 0 or 1, depending on the fact that f_i^y and f_{i+1}^y correspond to the same face of G or not. Since

$|s_i^*| \geq 0$ and, if $f_i^y \neq f_{i+1}^y$, $|s_i^*| \geq 1$, we have a contradiction in both the cases.

If μ_i is an R-node, then $w(f_i^y, f_{i+1}^y)$ is the length of the shortest path between f_i^y and f_{i+1}^y on the dual of $skel(\mu_i)$. Since both f_i^y and f_{i+1}^y belong to $skel(\mu_i)$ then, by Lemma 3.5, the length of such a shortest path is the minimum number of skip operations to move the external face from f_i^y to f_{i+1}^y, which contradicts the hypothesis that $|s_i^*| < w(f_i^y, f_{i+1}^y)$ and, hence, that $|s^*| < |s|$.

We now perform an analysis of the computational complexity of Algorithm SkipOnly. The SPQR-tree $\mathcal{T}$ [DT96b] and its labeling can be computed in linear time. The allocation trees $\mathcal{T}_1$ and $\mathcal{T}_2$ and $sp(f_1, f_2)$ can be also easily computed in linear time. Also, graph $Track(f_1, f_2)$ is constructed by computing at most 4 shortest paths for each R-node in $sp(f_1, f_2)$; since the sum of the number of virtual edges in the skeletons of nodes of $\mathcal{T}$ is $O(n)$, we have that $Track(f_1, f_2)$ can be computed in linear time. Finally, the simple level-structure of $Track(f_1, f_2)$ allows to compute the weighted shortest path from f_1 to f_2 in linear time. Namely, when considering a node f_{i+1}^l (f_{i+1}^r, respectively) of level $i+1$, we have already computed the weighted shortest paths from f_1 to nodes f_i^l and f_i^r of level i. Also, for each of f_i^l and f_i^r, we have only two possibilities to get to f_{i+1}^l (f_{i+1}^r, respectively), that is, either by using the vertical edge connecting them or by using the vertical edge connecting to f_{i+1}^r (f_{i+1}^l, respectively) and then traversing the horizontal edge (f_{i+1}^l, f_{i+1}^r) of weight 1. Hence, for each of the two nodes of level $i+1$, we perform a constant number of computations, which proves the statement. □

3.5 Linearity of the Case without P-nodes

In this section we show that, if $\mathcal{T}$ does not contain any P-nodes, the problem of computing $\mathcal{FS}(\langle\Gamma_1, f_1\rangle, \langle\Gamma_2, f_2\rangle)$ can be solved in linear time. For simplicity, the algorithm described in this section only considers a subset of all the possible flip operations. Namely, given an S-node μ, although a legitimate flip operation might concern the split components of any split pair of μ, we only consider flip operations concerning split components of maximal split pairs of μ. Intuitively, this corresponds to flipping either a single neighbor ν of μ or all the neighbors of μ except for ν. At the end of the section we handle the general case.

In order to compute $\mathcal{FS}(\langle\Gamma_1, f_1\rangle, \langle\Gamma_2, f_2\rangle)$ when $\mathcal{T}$ does not contain any P-nodes, we first assign a label in $\{\texttt{turned}, \texttt{unturned}\}$ to each node μ of $\mathcal{T}$, where the label assigned to a node μ indicates whether some transformations are needed on the skeleton of μ in order to obtain Γ_2 from Γ_1, or not.

- If μ is a Q-node, it is labeled `unturned`.
- If μ is an R-node, it is labeled `unturned` if the Boolean value assigned to μ in the two labelings representing Γ_1 and Γ_2 is the same, and `turned` otherwise.
- If μ is an S-node, it is labeled `unturned` (`turned`) if the majority of its adjacent R-nodes is `unturned` (`turned`). In case of a tie, we give μ an arbitrary label, unless μ is an internal node of the skip path *sp*. In this case, we give μ a label that is different from one of its adjacent nodes in *sp*. Observe that both of such nodes are R-nodes, since two S-nodes can not be adjacent in $\mathcal{T}$ and Q-nodes can not be in *sp*, because they do not have any internal faces.

Second, we suitably extend the labeling {`turned`, `unturned`} from the nodes to the edges of $\mathcal{T}$. If an edge e is incident to a Q-node, then it is labeled `unturned`. Otherwise, e is labeled `unturned` (`turned`) if its incident nodes have the same label (different labels).

Consider any edge $e = (\mu, \nu)$ of $\mathcal{T}$. Such an edge identifies a split pair of G which, in its turn, identifies two maximal split components $G_1 = \mathit{pertinent}(\mu, e(\nu|\mu))$ and $G_2 = \mathit{pertinent}(\nu, e(\mu|\nu))$, since $\mathcal{T}$ does not contain any P-nodes. By definition, the effect of performing a flip of G_1 (resp. G_2) is to change the labels of all the nodes of the subtree of $\mathcal{T}$ rooted at μ and not containing ν (resp. rooted at ν and not containing μ). Hence, e is the only edge of $\mathcal{T}$ whose label is changed by such a flip operation. It follows that, if e is labeled as `turned`, any minimum sequence of flips that transforms Γ_1 into Γ_2 contains either $\mathit{flip}(\langle \Gamma', f' \rangle, G_1)$ or $\mathit{flip}(\langle \Gamma'', f'' \rangle, G_2)$, for some suitable Γ', Γ'', f', and f''. Hence, the number of `turned` edges of $\mathcal{T}$ corresponds to the minimum number of flips that must be performed on Γ_1 in order to obtain Γ_2.

As in the fixed combinatorial embedding case, when the intersection of the two allocation trees $\mathcal{T}_1$ and $\mathcal{T}_2$ of f_1 and f_2 is non-empty, the problem has a trivial solution. In fact, since in this case there exists at least one node μ such that f_1 and f_2 belong to $\mathit{skel}(\mu)$, there is no flip that can help to reduce the number of skips. Hence, the trivial algorithm that first performs all the flips to transform Γ_1 into Γ_2 and then performs all the skips to move the obtained external face f_3 to f_2 uses $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ operations.

When the intersection of the two allocation trees is empty, we have to take into account the fact that a flip operation may modify the distance between any two faces not belonging to the same skeleton, hence modifying the number

of needed skips to move from one to the other. Therefore, in order to compute $\mathcal{FS}(\langle\Gamma_1, f_1\rangle, \langle\Gamma_2, f_2\rangle)$, we have to consider the case in which flip and skip operations are alternated.

Consider, for an example, the starting embedding $\langle\Gamma_1, f_1\rangle$ of Fig. 3.5(a) and the target embedding $\langle\Gamma_2, f_2\rangle$ of Fig. 3.5(b). In Γ_1 we have that f_2 is adjacent to f_1, while in Γ_2 the minimum number of skips to reach f_2 from f_1 is 5 (observe that an example where the distance is k can be easily constructed by adding $O(k)$ vertices and edges). Hence, it would be possible to save 4 (in general, $k-1$) skip operations by moving the external face to f_2 before performing the flips needed to transform Γ_1 into Γ_2. On the other hand, such an operation would make the rigid component R_4 not possible to be flipped, since it would contain all the edges of the new external face f_2. However, when a flip operation of a component R_i, with $1 \leq i \leq 4$, that involves R_4 is needed in order to obtain an embedding Γ_*, it is possible to perform such a flip operation on the unique component R_j, with $1 \leq j \leq 4$ and $i \neq j$, that shares its split pair with R_i, so obtaining an embedding $\overline{\Gamma_*}$ that is equal to Γ_*, but for a reversal of the adjacency lists of all the vertices. Hence, when all the flips are performed, the combinatorial embedding Γ_3 that is obtained is such that either $\Gamma_3 = \Gamma_2$ or $\Gamma_3 = \overline{\Gamma_2}$, depending on the fact that the number of flips involving the component containing f_2 is even or odd. In this example, 3 flips involving R_4 are needed to obtain Γ_2 from Γ_1, that is, the flip of R_2, R_3, and R_4, and hence the final embedding is $\overline{\Gamma_2}$. In order to obtain Γ_2, it is possible to perform a flip of the whole graph G around a simple edge e incident to f_2 since, by Property 3.3, we have that $\mathit{flip}(\langle\overline{\Gamma_2}, f_2\rangle, G \setminus e) = \langle\Gamma_2, f_2\rangle$. Therefore, the total number of flip operations is equal to the number of flips needed to transform Γ_1 into Γ_2 plus one. Hence, the total number of operations that are saved by first moving the external face to f_2 and then transforming Γ_1 into Γ_2 is 3 (in general, at least $k-2$).

An analogous example where first performing all the skips and then performing all the flips leads to a solution with a non-optimal number of operations can be easily constructed by considering the embedding of Fig. 3.5(b) as the starting one and the embedding of Fig. 3.5(a) as the target one.

Based on these observations, we propose an algorithm, called NoParallel, to compute $\mathcal{FS}(\langle\Gamma_1, f_1\rangle, \langle\Gamma_2, f_2\rangle)$ when $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$ and $\mathcal{T}$ does not contain any P-nodes. Such an algorithm is similar to Algorithm SkipOnly and its pseudo-code is given in Algorithm 2. The main difference is on the assignment of the weights to the edges of $\mathit{Track}(f_1, f_2)$, which have to take into account the possibility of alternating flips and skips in order to reduce the total number of operations. Namely, consider two nodes μ_i and μ_{i+1} of the skip path $sp(f_1, f_2)$

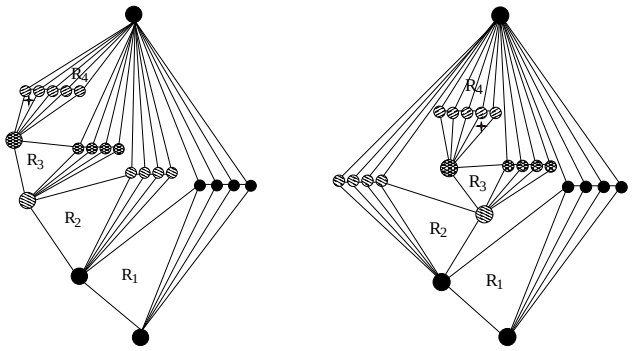


Figure 3.5: An example of morphing in which it is possible to save 3 operations by performing some skips before the flips.

which are adjacent through a `turned` edge e, and consider a skip operation on μ_{i+1}. Such an operation has the effect of transferring the external face from f^l_{i+1} to f^r_{i+1}, or vice versa. The same effect would be obtained by flipping μ_{i+1} with respect to μ_i. Therefore, we set to 0 the weight of the horizontal edge linking f^l_{i+1} to f^r_{i+1} in graph $\mathit{Track}(f_1, f_2)$ and we call *shortcut* such an edge. Using a shortcut in the shortest path from f_1 to f_2 corresponds to the possibility of performing a flip in advance to save a skip operation, while a shortcut that is not in the shortest path corresponds to a flip that has to be performed at the end of the computation.

First observe that all the flips that do not involve nodes on the skip path can be performed independently at each step of the computation, since none of these nodes will contain all the edges of the external face during the morphing. We decide to perform such flips as the first step of the algorithm. As for Algorithm 1, compute a weighted shortest path p on $\mathit{Track}(f_1, f_2)$ from f_1 to f_2. First, perform the flip operations corresponding to the shortcuts that are traversed by p, while the external face is still f_1. Second, perform all the skip operations corresponding to the edges of sp. Finally, perform the flip operations corresponding to all the other `turned` edges (the shortcuts that are not traversed by p), while the external face is f_2.

Regarding the flips that are performed before the skips, we have to consider the fact that the weighted shortest path between f_1 and f_2 on $\mathit{Track}(f_1, f_2)$ does

not take into account the label of the last node μ_k of the skip path $sp(f_1, f_2)$. Such a matter is considered in the computation of $\mathcal{FS}(\langle\Gamma_1, f_1\rangle, \langle\Gamma_2, f_2\rangle)$ by suitably selecting one of the weighted shortest paths, as follows.

Suppose that μ_k is labeled `turned` (`unturned`). Also, suppose that a weighted shortest path p_1 from f_1 to f_2 uses an even (odd) number of shortcuts. If the flip operations corresponding to such shortcuts are performed in advance, while f_1 is the external face, the embedding of μ_k is reversed an even (odd) number of times, i.e., μ_k ends up with a `turned` label. Hence, in order to obtain Γ_2, according to Property 3.1, we would need to perform a final flip operation with respect to any edge incident to f_2. However, in this case, if a weighted shortest path p_2 from f_1 to f_2 exists with the same cost of p_1 and traversing an odd (even) number of shortcuts, using p_2 would make it possible to save the last flip. Therefore, we compute two weighted shortest paths from f_1 to f_2 using an even and an odd number of shortcuts, respectively. Then, if they have different cost we choose the shorter one, while if they have the same cost we select the one that makes the label of μ_k `unturned`. Hence, when all of such flips have been performed, μ_k can be labeled either `turned` or `unturned`.

Regarding the flips that are performed after all the skips, we recall that, in order to preserve the mental map of the user, a flip operation on a component G_1 is permitted only if G_1 does not contain all the edges of the external face. Further, we observe that, if μ_k is labeled `turned`, then at least one of such flips must involve μ_k, which contains all the edges of the current external face f_2 and hence can not be flipped. However, in this case, it is possible to exploit the fact that, when a flip operation of a component μ is needed in order to obtain an embedding Γ_*, it is possible to perform such a flip operation on the unique component ν sharing its split pair with μ, obtaining an embedding $\overline{\Gamma_*}$ that is equal to Γ_*, but for a reversal of the adjacency lists of all the vertices.

Hence, when both the flips before the skips and the ones after the skips are performed, the combinatorial embedding Γ_3 that is obtained is such that either $\Gamma_3 = \Gamma_2$ or $\Gamma_3 = \overline{\Gamma_2}$, depending on the fact that the total number of flips involving μ_k is even or odd. However, when $\Gamma_3 = \overline{\Gamma_2}$, it is possible to obtain Γ_2 by performing a flip of the entire graph G around one edge e incident to f_2, since, by Property 3.3, we have that $\mathit{flip}(\langle\overline{\Gamma_2}, f_2\rangle, G \setminus e) = \langle\Gamma_2, f_2\rangle$.

Finally observe that, if the two weighted shortest paths traversing an even and an odd number of shortcuts, respectively, have the same cost, it is always possible to obtain Γ_2 at the end of the computation without the need of performing the last flip of the whole graph around e.

Theorem 3.3 *Let G be a biconnected planar graph, and let $\langle\Gamma_1, f_1\rangle$ and $\langle\Gamma_2, f_2\rangle$*

be two planar embeddings of G. Let $\mathcal{T}$ be the SPQR-tree of G. If $\mathcal{T}$ does not contain any P-nodes, then $\mathcal{FS}(\langle\Gamma_1, f_1\rangle, \langle\Gamma_2, f_2\rangle)$ can be computed in linear time.

Proof: Apply algorithm NoParallel. Let s be the number of operations performed on G. First observe that, since each flip operation modifies the Boolean value of all the nodes of a subtree of $\mathcal{T}$, we have that only one edge of $\mathcal{T}$ has its label modified by this operation. Hence, the number of flip operations performed on G is greater than, or equal to, the number of `turned` edges in $\mathcal{T}$ plus (possibly) one flip of the whole graph that has to be performed at the end of the computation (see Property 3.3).

First observe that the flips involving nodes that are not on the skip path can not be saved by any skip operation, since moving the external face to these nodes in order to save one flip would imply at least two unnecessary skips, one to reach it and one to go back to the skip path. Hence, all of such flips have to be performed at some point. Moreover, the nodes involved in these flips will not contain all the edges of the external face at any step of the computation and hence they can be performed as the first step of the algorithm without affecting the optimality of the solution.

Consider an horizontal edge e of weight 1, that is, corresponding to an `unturned` edge. Since any flip operation can reduce the skip-distance between f_1 and f_2 of at most 1, it is not useful to perform a flip operation on a component corresponding to e, because in this case another flip operation would be needed to restore the combinatorial embedding. Hence, any optimal sequence of operations that morphs Γ_1 into Γ_2 must not contain any flip operations on the `unturned` edges.

Consider an horizontal edge $e = (f_i^l, f_i^r)$ of weight 0 (a shortcut), that is, corresponding to a `turned` edge, belonging to the suitably selected weighted shortest path in $\mathit{Track}(f_1, f_2)$ from f_1 to f_2. The fact that the weighted shortest path traverses e means that, in the optimal sequence of operations induced by this path, a skip of the whole component corresponding to e would be needed to move the external face from f_i^l to f_i^r, or vice versa. However, if the flip corresponding to e, which is needed anyhow, is performed before the skips, the roles of f_i^l and f_i^r are switched and hence the skip of the component corresponding to e is no longer needed, which results in saving one operation. Therefore, we have that any optimal sequence of operations must perform all the flip operations corresponding to shortcuts that are traversed by the suitable selected weighted shortest path before the skip operations involving such components.

Consider an horizontal edge $e = (f_i^l, f_i^r)$ of weight 0 (a shortcut), that is, corresponding to a `turned` edge, not belonging to the selected weighted

Algorithm 2 NoParallel

Require: A biconnected planar graph G and two of its planar embeddings $\langle \Gamma_1, f_1 \rangle$ and $\langle \Gamma_2, f_2 \rangle$.

Ensure: The minimum length sequence $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ of flip and skip operations to obtain $\langle \Gamma_2, f_2 \rangle$ from $\langle \Gamma_1, f_1 \rangle$.

Prepocessing Phase

Compute the SPQR-tree $\mathcal{T}$ of G
Label the nodes of $\mathcal{T}$ according to Γ_1
Label the nodes of $\mathcal{T}$ according to Γ_2
Compute the allocation trees $\mathcal{T}_1$ and $\mathcal{T}_2$ of f_1 and f_2, respectively
Label the nodes of $\mathcal{T}$ as `turned` or `unturned` as described in Sect. 3.5
Label the edges of $\mathcal{T}$ as `turned` or `unturned` as described in Sect. 3.5

Computation Phase

Compute the skip path $sp(f_1, f_2)$ between $\mathcal{T}_1$ and $\mathcal{T}_2$
Construct the track graph $Track(f_1, f_2)$, assigning weight 0 to an horizontal edge (f_i^l, f_i^r) if the edge of $\mathcal{T}$ connecting μ_i and μ_{i-1} is `turned` and weight 1 otherwise
Compute two weighted shortest paths p_{even} and p_{odd} on $Track(f_1, f_2)$ from f_1 to f_2 traversing an even and an odd number of shortcuts, respectively
if cost(p_{even}) < cost(p_{odd}) **then**
  $p = p_{even}$
else
  if cost(p_{even}) > cost(p_{odd}) **then**
    $p = p_{odd}$
  else
    Select the one of $p = p_{even}$ or $p = p_{odd}$ that makes the label of μ_k `unturned`
  end if
end if

Execution Phase

Perform all the flips that do not involve nodes on the skip path
Perform all the flips corresponding to the shortcuts traversed by p
Perform all the skips corresponding to the edges of p to move the external face from f_1 to f_2
Perform all the flips corresponding to the shortcuts not traversed by p on the components not containing all the edges incident to f_2
if label(μ_k) == `turned` **then**
  Perform a skip of the whole graph around an edge incident to f_2
end if

shortest path p in $\mathit{Track}(f_1, f_2)$. Suppose, by contradiction, that there exists a sequence s^* of operations such that e is flipped before the skips, and that $|s^*| < |s|$. Since s^* contains the same flip operations as s, then s^* must contain a smaller number of skips. Hence, whereas the external face must traverse all the nodes of $sp(f_1, f_2)$, by Property 3.5, sequence s^* would identify a path in $\mathit{Track}(f_1, f_2)$ that is shorter than p, which is a contradiction. Therefore, we have that any optimal sequence of operations must perform all the flip operations corresponding to shortcuts that are not traversed by the suitable selected weighted shortest path after the skip operations involving such components.

Since a flip operation of the whole graph reverses the adjacency lists of all the vertices but does not modify the number of `turned` edges, any optimal sequence will perform at most one of such flips. Moreover, if two shortest paths exist traversing an even and an odd number of shortcuts, respectively, such a flip can be saved by choosing the shortest path which ends with an `unturned` label on the last node of the skip path.

Finally, since each edge of $\mathit{Track}(f_1, f_2)$ is weighted with the length of the shortest path on the skeleton of the corresponding component, we have that no other sequence of skips can outperform the one that is computed with Algorithm NoParallel, when the savings permitted by the flips are considered.

Hence, we have that algorithm NoParallel computes $\mathcal{FS}(\langle\Gamma_1, f_1\rangle, \langle\Gamma_2, f_2\rangle)$.

The fact that NoParallel can be executed in linear time can be shown with the same argumentation presented in the proof of Theorem 3.2. In fact, the computation of the two shortest paths traversing an even and an odd number of shortcuts can be performed in linear time since, for each level i, we have to perform a choice for the two nodes f_i^l and f_i^r that is based on the choices made for the two nodes f_{i-1}^l to f_{i-1}^r of level $i-1$ and on the weight of the four edges that connect the vertices of the two levels, which results in a constant number of operations for each level. □

Now we show how to modify Algorithm NoParallel in order to handle the general case in which a flip operation may concern the split component induced by any split pair of an S-node μ. Intuitively, this corresponds to allow the flip of an arbitrary number of consecutive neighbors of μ with one single operation. The idea is to relax the constraint that two S-nodes can not be adjacent in the SPQR-tree of G. Namely, for any maximal sequence $\sigma_i = \nu_1, \nu_2, \ldots, \nu_k$ of consecutive nodes with the same label adjacent to μ, we add an S-node μ_i adjacent to μ and move σ_i from the adjacency list of μ to the one of μ_i. The label of μ_i is the same as the one of all the nodes on σ_i. The label of μ is computed as for Algorithm NoParallel.

3.6 Fixed-Parameter Tractability of the General Case

In Sect. 3.3, we showed that the problem of transforming $\langle \Gamma_1, f_1 \rangle$ into $\langle \Gamma_2, f_2 \rangle$ with the minimum number of flip and skip operations is NP-complete when G is an arbitrary biconnected planar graph. In this section, we study the fixed-parameter tractability of the problem when the structure of G is of limited complexity.

Let $\mathcal{T}$ be the SPQR-tree of a biconnected planar graph G and let $\langle \Gamma_1, f_1 \rangle$ and $\langle \Gamma_2, f_2 \rangle$ be two planar embeddings of G. We present an algorithm to compute a sequence of $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ flip and skip operations that transforms $\langle \Gamma_1, f_1 \rangle$ into $\langle \Gamma_2, f_2 \rangle$ in $O(n^2 \times 2^{k+h})$ time, where k and h are two parameters that describe the arrangement of the P-nodes of $\mathcal{T}$ and their relationships with the S-nodes.

We first sketch out how to handle P-nodes, which are responsible for the NP-hardness of the general problem, with a fixed-parameter tractability approach. Recall that the embedding of the skeleton of a P-node μ_P is described in the labelings of $\mathcal{T}$ representing Γ_1 and Γ_2 by two circular sequences σ_1 and σ_2, respectively, of its adjacent virtual edges. As shown in the proof of Theorem 3.1, the problem of morphing σ_1 into σ_2 with the minimum number of flips is equivalent to the *sorting by reversal* problem (SBR), which has been proved to be NP-hard in both cases of linear and circular sequences [Cap97, SSL03]. In fact, sorting virtual edges is equivalent to sorting integer numbers, where a flip of l contiguous edges corresponds to a reversal of l contiguous elements of the sequence of integer numbers.

The fixed-parameter approach is based on the fact that the SBR problem can be solved in polynomial time, both in its linear and in its circular formulation, when each number is given a sign and the reversal of l contiguous elements of a sequence also changes their signs [KST97, TBS07, MWD00].

Indeed, when all the virtual edges adjacent to a P-node correspond to components that have to be suitably "flipped" while being reordered, that is, they can be provided with a sign, then the problem of morphing σ_1 into σ_2 can be modeled as an instance of the signed SBR problem, hence admitting a polynomial-time solution. For example, if all the nodes adjacent to a P-node are R-nodes, which can always be provided with a sign, then the minimum number of flips to sort them can be computed in polynomial time. Unfortunately, some virtual edges, as for example those corresponding to paths, used in the NP-hardness proof of Sect. 3.3, do not need to be flipped in a specific way.

In order to exploit the polynomial-time solvability of the signed SBR prob-

lem while taking into account the fact that not all the virtual edges can be assigned a sign without affecting the optimality of the solution, we assume that the number of such virtual edges around each P-node is limited by a parameter k. Then, we conventionally assign to such k virtual edges all the combinations of signs, and we apply 2^k times the signed SBR polynomial-time algorithm. In fact, there exists an assignment of signs such that the minimum number of reversals to order the corresponding signed sequence is equal to the minimum number of reversals to order the original mixed signed/unsigned sequence [AA04].

As in the previous cases, we focus on the case in which $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$, that is the most complex. The case when $\mathcal{T}_1 \cap \mathcal{T}_2 \neq \emptyset$ can be tackled with similar techniques. The algorithm is described in detail below.

In order to compute $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$, each node of $\mathcal{T}$ is labeled as `turned`, `unturned`, or `neutral`, as described in the following. Intuitively, labels `turned` and `unturned` represent the sign of a component, while label `neutral` is assigned to components that can not be provided with a sign. First, we order the nodes of $\mathcal{T}$ based on their distance from the skip path *sp*. Then, starting from the farthest ones, we label the nodes that are not in *sp* with the strategy described below. Finally, we label the nodes of *sp* with a different strategy.

Now we describe how to label the elements that are not on the skip path. Consider the current unlabeled node μ not in *sp*. Observe that all the nodes adjacent to μ, with the exception of the node that links μ to *sp*, have already been assigned a label.

- If μ is an R-node, we label μ based on its embedding, as described in Section 3.5 for Algorithm NoParallel. Observe that such a label is either `turned` or `unturned`, but it is never `neutral`.
- If μ is a Q-node, we label μ `neutral`.
- If μ is an S-node, we assign μ the label of the majority of its non-neutral labeled adjacent nodes. In case of a tie, we label μ `neutral`.
- If μ is a P-node, denote by σ_1 and σ_2 the two circular sequences associated with μ in the labelings of $\mathcal{T}$ representing Γ_1 and Γ_2, respectively. When labeling μ, at the same time we also compute the minimum number of flips that are needed to transform σ_1 into σ_2.
  First observe that, since the external face traverses only the nodes of $\mathcal{T}$ belonging to the skip path, by Property 3.5, at each step of the computation all the edges incident to the external face are contained into

pertinent$(\nu, e(\mu|\nu))$, where ν is the node of $\mathcal{T}$ that links μ to *sp*. Hence, node ν can never be part of a flip/reversal during the sorting of the circular sequence of the nodes adjacent to μ, which implies that σ_1 and σ_2 are actually linear sequences, as far as only flip operations are concerned. Namely, denote by σ'_1 and σ'_2 the two linear sequences obtained from σ_1 and σ_2, respectively, by removing the virtual edge $e(\nu|\mu)$ and starting with the virtual edge that follows $e(\nu|\mu)$ in σ_1.
Let k be the number of `neutral` elements of σ'_1 and σ'_2. We assign all possible combinations of `turned` and `unturned` values to such elements, and compute 2^k times the linear signed SBR distance d from σ'_1 to σ'_2. Also, we compute the analogous distance $\overline{d}$ from σ'_1 to $\overline{\sigma}'_2$, where $\overline{\sigma}'_2$ is the sequence obtained from σ'_2 by completely reversing the order and by changing the signs of all the elements of the sequence. More precisely, $\overline{\sigma}'_2$ is obtained as the result of a flip/reversal of the whole sequence σ'_2. If $d < \overline{d}$ ($d > \overline{d}$, $d = \overline{d}$, respectively) we assign μ the label `unturned` (`turned`, `neutral`, respectively).

Now we describe how to assign labels to the elements of the skip path *sp*$= \mu_1, \mu_2, \ldots, \mu_k$ from $\mathcal{T}_1$ to $\mathcal{T}_2$. We construct a labeling such that nodes in *sp* are never labeled `neutral`.

First observe that no Q-node μ can be on *sp*, as *pertinent*$(\mu, e(\nu|\mu))$, where ν is the only node of $\mathcal{T}$ adjacent to μ, does not contain any internal faces. The labels of the R-nodes are assigned based on their embeddings, as in Sect. 3.5.

Labels to the S-nodes μ_S are assigned as in Sect. 3.5, that is, by breaking a possible tie in the number of `turned` and `unturned` neighbors in such a way that the label of μ_S is different from the label of at least one of its neighbors in *sp*. Since R-nodes have always a sign and two S-nodes can not be adjacent in $\mathcal{T}$, the only case to consider is when both of its neighbors in *sp* are P-nodes, whose sign still depends on its neighbors, which creates a mutual dependence. Such a problem can be extended to the case in which we have an alternated sequence of S- and P-nodes on *sp*. Hence, we assume that the number of P-nodes in *sp* is limited by a parameter h and we compute the fixed-parameter tractable solution based also on this parameter. Namely, if we have h P-nodes in *sp*, we consider for them all the 2^h combinations of the two labels `turned` and `unturned` and, for each of them, we compute the labels of the adjacent S-nodes in *sp* and we perform the following computation.

As in Algorithm NoParallel, we extend the labeling from the nodes to the edges of $\mathcal{T}$. In particular, an edge is labeled `turned` if its incident nodes have different labels and none of them is a P-node, otherwise it is `unturned`.

We construct a *weighted track graph* $Track(f_1, f_2)$ as in Algorithm NoParallel, where P-nodes were not present. Here we describe how to set the weights of the edges exiting nodes f_i^l and f_i^r corresponding to a P-node μ_i in *sp*. All other weights are set as described in Sect. 3.5.

The weight of an horizontal edge for a P-node is 1, representing the possibility to skip the whole parallel component with one single operation.

In order to set the weight of a vertical edge for a P-node, we have to consider that, since the external face moves along the nodes of the skip path *sp*, there is no neighbor ν_i of μ_i such that $pertinent(\nu, e(\mu_i, \nu))$ contains all the edges incident to the external face during all the computation. Therefore, it is not possible to remove one of the nodes from the sequence and consider it as linear, as done for the P-nodes that are not in *sp*.

However, it is still possible to represent the fact that the current external face is a face of the skeleton of μ_i by means of two linear sequences, by cutting the circular sequence in such a way that the virtual edge representing the component μ_{i-1}, that precedes μ_i in *sp*, is the first and the last element, respectively, of the two obtained linear sequences.

More precisely, denote by $\sigma_{1,i}$ and $\sigma_{2,i}$ the two circular sequences representing the embedding of μ_i in Γ_1 and Γ_2, respectively. From $\sigma_{1,i}$ we obtain the linear sequence $\sigma_{1,i}^l$ ($\sigma_{1,i}^r$) ending with (starting with, respectively) the virtual edge corresponding to μ_{i-1}. Intuitively, sequence $\sigma_{1,i}^l$ ($\sigma_{1,i}^r$) corresponds to the configuration of the parallel component when the external face is f_i^l (f_i^r). Analogously, from $\sigma_{2,i}$ we obtain the linear sequence $\sigma_{2,i}^l$ ($\sigma_{2,i}^r$) ending with (starting with, respectively) the virtual edge corresponding to μ_{i+1}.

Our aim is to set the weight of each vertical edge (f_i^s, f_{i+1}^t), with $s, t \in \{l, r\}$, as the minimum number of operations needed to transform $\sigma_{1,i}^s$ into $\sigma_{2,i}^t$. Observe that, when the external face is moved from f_i^s to another face f of $skel(\mu_i)$ in Γ_1, we obtain a new linear sequence $\sigma_{1,i}^*$ corresponding to a different cut of the same circular order as $\sigma_{1,i}^s$. Namely, σ_1^* is obtained from σ_1 by cutting it between the two virtual edges adjacent to f. Hence, when computing the minimum number of operations needed to transform $\sigma_{1,i}^s$ into $\sigma_{2,i}^t$, we have to consider the possibility of first transforming $\sigma_{1,i}^s$ into a different linear sequence $\sigma_{1,i}^*$ with the same circular order, that can be done with one skip operation, and then transforming $\sigma_{1,i}^*$ into $\sigma_{2,i}^t$ with the minimum number of flips, that can be done by applying the signed SBR algorithm. In order to do this, observe that all the nodes adjacent to μ_i in $\mathcal{T}$ are labeled as `turned`, `unturned`, or `neutral`. Let k be the number of nodes adjacent to μ_i and labeled `neutral`. As described above, we consider all possible assignments of `turned`

and `unturned` values to such nodes, and we compute 2^k times the linear signed SBR distance from $\sigma^*_{1,i}$ to $\sigma^t_{2,i}$. The weight of the vertical edge (f^s_i, f^t_{i+1}) is the minimum of such $n_i \times 2^k$ values, where n_i is the number of faces of $skel(\mu_i)$, which is equal to the number of nodes adjacent to μ_i in $\mathcal{T}$.

The remaining part of the algorithm strictly follows the lines of Algorithm NoParallel. Namely, we compute a weighted shortest path from f_1 to f_2 in $Track(f_1, f_2)$ and, based on such a path, we decide the sequence of skip and flip operations to be performed. Again, if $Track(f_1, f_2)$ admits more than one weighted shortest path, we choose among such paths taking into account the number of shortcuts traversed, corresponding to flip operations that are convenient to be performed in advance.

Here we analyze the computational complexity of the algorithm. All the operations, except for those involving P-nodes, can be performed in linear time, as stated in Section 3.5.

For each P-node μ_i not belonging to the skip path, the computation of the minimum sequence of flips needed to transform $\sigma_{1,i}$ into $\sigma_{2,i}$ can be performed in $O(n_i \times 2^k)$ time, where n_i is the number of neighbors of μ_i in $\mathcal{T}$. Observe that computing the minimum SBR distance can be done in linear time [BMY01], while actually finding the sequence of operations that yields that minimum can be done in time $O(n_i^{\frac{3}{2}}\sqrt{log(n_i)})$ time [TBS07]. Hence, when considering the 2^k possible assignments, we only compute the minimum SBR distance and then, when the optimal assignment has been found, we perform the algorithm for finding the actual sequence of flips.

For each P-node μ belonging to sp, the computation of the minimum sequence of flips needed to transform $\sigma^s_{1,i}$ into $\sigma^t_{2,i}$ can be performed in $O(n_i^2 \times 2^k)$. Namely, we have to consider the 2^k assignments of signs to the k `neutral` neighbors of μ_i, which requires $O(n_i \times 2^k)$ time, and the possibility to transform $\sigma^s_{1,i}$ into $\sigma^t_{2,i}$ by first moving the external face to each of the n_i faces of $skel(\mu_i)$ in Γ_1 and then performing the computation of the signed linear SBR distance in linear time. Since such a computation has to be performed for each of the 2^h assignments of labels to the h P-nodes of sp, the total computational complexity of the algorithm is $O(2^h \times \sum_{i=1}^{h}(n_i^2 \times 2^k))$, which is equal to $O(n^2 \times 2^{k+h})$, since the total number of neighbors of all the P-nodes is less than, or equal to, the total number of edges of $\mathcal{T}$, that is $O(n)$.

Based on the above discussion we have:

Theorem 3.4 *Let G be a biconnected planar graph and let $\langle \Gamma_1, f_1 \rangle$ and $\langle \Gamma_2, f_2 \rangle$ be two planar embeddings of G. Let $\mathcal{T}$ be the SPQR-tree of G, let k be the*

maximum number of neutral S-nodes adjacent to a P-node in $\mathcal{T}$, and let h be the number of P-nodes in the skip path $\mathrm{sp}(f_1, f_2)$. *If both flip and skip operations are allowed, then* $\mathcal{FS}(\langle\Gamma_1, f_1\rangle, \langle\Gamma_2, f_2\rangle)$ *can be computed in* $O(n^2 \times 2^{k+h})$ *time.*

3.7 Conclusions

Preserving the user mental map while coping with ever-changing information is a common goal of the Graph Drawing and the Information Visualization areas. The information represented, in fact, may change with respect to three different levels of abstraction: (i) structural changes may modify the graph that the user is inspecting; (ii) topological changes may affect the way the same graph is embedded on the plane; and (iii) drawing changes may map the same embedded graph to differently positioned graphic objects.

A large body of literature has been devoted to structural changes, addressing the representation models and techniques in the so-called dynamic and on-line settings. Also, much research effort has been devoted to manage drawing changes, where the target is to preserve the mental map by morphing the picture while avoiding intersections and overlappings. On the contrary, to our knowledge, no attention at all has been devoted to topological changes, that is, changes of the embedding of a graph in the plane. In [KL08], Kobourov and Landis consider the morphing of two planar drawings of a maximal planar graph on the sphere, with the intent of studying the effects of changing the external face in the drawing while maintaining the combinatorial embedding.

In this chapter we addressed the topological morphing problem. Namely, the problem of morphing a topology into another one with a limited number of changes. Many open problems are left. (1) Primitives. We considered two topological primitives, called flip and skip. It would be important to enrich such a set with other operations that can be considered "natural" for the user's perception. (2) Connectivity. It is easy to extend the results presented in Sect. 3.4 to simply connected graphs. However, the other presented results are deeply related to biconnectivity. There is a lot of space here for further investigation. (3) We gave the same weight to the operations performed during the morphing. However, other metrics are possible. For example, one could weight an operation as a non-decreasing function of the moved edges or of the thickness of the moved component.

As a final remark we underline how usually the Computational Biology field looks at Graph Drawing as a tool. In this case it happened the opposite. In fact, Theorems 3.1 and 3.4 exploit Computational Biology results.

Chapter 4

Testing Planarity of Partially Embedded Graphs

In this chapter[1], we consider a problem strongly related to the classical planarity testing problem, as we study the planarity testing in a setting in which the planar embedding that we aim to compute has to respect some further constraints given as part of the input. Namely, given a planar graph and a planar embedding of one of its subgraphs, we ask whether the embedding of such a subgraph can be extended to a planar embedding of the entire graph.

Observe that this problem is a generalization of the planarity testing problem, as solving an instance in which H does not contain any edge (or in which the edge-set of H induces a set of paths) is equivalent to solving an instance of the planarity testing with G as input.

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, that is, we show a characterization of the partially embedded graphs that admit an embedding extension and a linear-time algorithm for testing embedding extension.

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

[1] The work presented in this chapter is part of a joint work with Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter, appeared in [ADF+10].

it, and argue that they are NP-hard. Also, we show how our algorithm can be applied to solve related Graph Drawing problems, as the one of embeddings two planar graphs on the same set of points when the edges shared by the two graphs have to be represented by the same curve and the embedding of one of the two graphs is fixed in advance.

4.1 Introduction

In this chapter we pose and study the question of planarity testing in a constrained setting, namely when a part of the input graph is already embedded 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 an embedding 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 [Wes92, Tam96, DT96b, Pou94], are not suitable to be used in this context.

The question of testing the planarity of partially embedded 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 [KS97], 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 [Fia03], 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 chapter. All the more so since this problem is known to be NP-hard [Pat06] 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., [TDB88, Tam98, Dor02, GKM08]. Unfortunately, none of those results can be exploited to solve the question we pose. Mohar [Moh99, JM05] 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 con-

nected graphs to be part of the input. It is readily seen that in this case the rotation schemes of the vertices do not fully describe the input. In fact, the relative position of vertices against cycles in the graph must also be considered. 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. 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 some 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.

Further, we study some problems that are a generalization of the planarity testing of partially embedded graphs, as for example the problem in which the maximum number of edges that can be reinserted in the partially drawn graph while maintaining planarity is requested, and we show that they are NP-hard.

Finally, we show that testing the planarity of partially drawn graphs is equivalent to the special case of the simultaneous embedding with fixed edges problem in which one of the two graphs has a fixed embedding. Hence, as a side result, we show that our algorithm solves this problem in linear time.

The chapter is organized as follows. In Section 4.2 we describe the terminology and list auxiliary topological lemmata. In particular, the combinatorial invariants of equivalent embeddings are introduced. In Section 4.3 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 4.4 is then devoted to describe technical details of the linear-time algorithm. In Sect. 4.5 we present some generalizations of the problem. In Sect. 4.6 we study how our techniques can be used to solve other Graph Drawing problems, as the special case of simultaneous embedding with fixed edges. Section 4.7 summarizes the results and lists some related open problems.

4.2 Notation and Preliminaries

In this section we introduce some notations and preliminaries.

First, we give a definition of embedding that is slightly different from the one used till now, since it has to take into account the relative positions of different connected component of a non-connected graph, that happens to be important in the problem considered in this chapter, but that is usually irrelevant for other type of problems. For example, in the planarity testing different connected components can be treated separately and then recomposed in any way, while in this case it is not possible. Namely, suppose that the part of the graph that has already been drawn contains a cycle C and a vertex v, belonging to a different connected component, that is drawn inside C. Then, such an information can not be encoded by simply specifying the rotation scheme of each vertex, and a further information is needed.

Let D be a planar drawing of a graph G. Visiting the (not necessarily connected) border of a face f of D 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. We say that two drawings are *equivalent* if they have the same rotation scheme for each vertex 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. 4.1 for an example):

Problem 4.1 (Partially Embedded Planarity (Pep)) *Given a triplet $(G, H, \mathcal{H})$ of two graphs G and H, with $H \subseteq G$, and a planar embedding $\mathcal{H}$ of H, does G admit a planar embedding whose restriction to H is $\mathcal{H}$?*

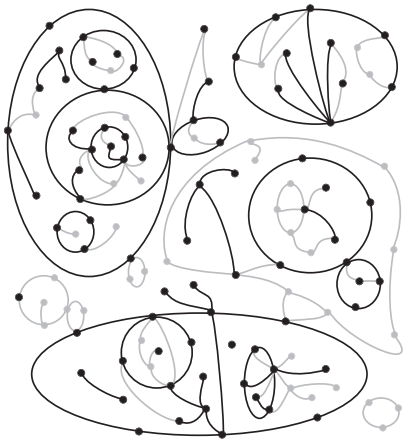


Figure 4.1: Embedding of a planar graph G whose restriction to H is $\mathcal{H}$. Vertices and edges in H are black; vertices and edges in $G \setminus H$ are grey.

Connectivity and data structures

Let $(G, H, \mathcal{H})$ be an instance of Pep. In the following we define some data structures that are widely used throughout the chapter.

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 *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.

Also, we will make deep use of BC-trees [HP66] and SPQR-trees [DT96b].

In the following we discuss the time complexity of constructing such data structures.

First, observe that a linear-time preprocessing can associate each edge of a planar graph with the unique connected component it belongs to, with the unique block it belongs to, and (given a planar embedding of the graph) with

the at most two faces it is incident to, and can associate each vertex of a graph with the unique connected component it belongs to.

The block-cutvertex tree of a connected planar graph and the SPQR-tree of a biconnected planar graph can be constructed in linear time [GM00].

The enriched block-cutvertex tree of a connected planar graph G is constructed by adding to the block-cutvertex tree of G each vertex v that is not a cutvertex of G and an edge between v and the only block it belongs to.

The block-face tree $\mathcal{BF}$ of a planar embedding $\mathcal{G}$ of a graph G can be constructed in linear time. Namely, for each edge e of G, let B_e be the unique block of G containing e and let f'_e and f''_e be the two faces of $\mathcal{G}$ adjacent to e (possibly $f'_e = f''_e$). Add edges (f'_e, B_e) and (f''_e, B_e) to $\mathcal{BF}$. When all the edges of G have been considered, the resulting multigraph $\mathcal{BF}$ has a linear number of edges. Remove multiple edges as follows. Root $\mathcal{BF}$ at any node and orient $\mathcal{BF}$ so that all the edges point toward the root. Remove all the edges, except for one, exiting from each node, thus obtaining $\mathcal{BF}$.

The component-face tree $\mathcal{CF}$ of a planar embedding $\mathcal{G}$ of a planar graph G can be constructed in linear time, analogously as for the block-face tree.

The vertex-face incidence graph $\mathcal{VF}$ of a planar embedding $\mathcal{G}$ of a graph G is constructed in linear time by processing faces of $\mathcal{G}$ one by one. In fact, for each face f, walk along the boundary of f and add to $\mathcal{VF}$ edges between f and the vertices on the boundary. To avoid multiple edges, we remember, for each vertex x, the last face f that has been connected to x in $\mathcal{VF}$.

Kowalik and Kurowski [KK03] have shown that for a given planar graph F and an integer k, it is possible to build in linear time a 'short-path' data structure, which allows to check in constant time whether two given vertices of F are connected by a path of length at most k, and return such a path if it exists. We will employ this data structure to search for paths of length 1 and 2 in our auxiliary graphs. Using this data structure, we can, e.g., determine in constant time whether two vertices share the same face in $\mathcal{H}$ (by finding a path of length two in the vertex-face incidence graph $\mathcal{VF}$) or whether they share the same block (by finding a path of length 2 in the enriched block-cutvertex tree).

Facial cycles and H-bridges

Let Γ be a planar drawing of a graph H (see Fig. 4.2.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. 4.2.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. 4.2.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 $\mathcal{H}$.

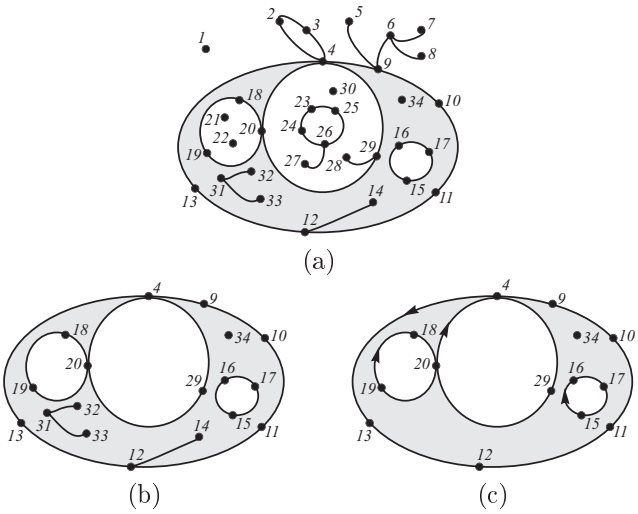


Figure 4.2: (a) A planar drawing of a graph G. The shaded region represents a face f of the drawing. (b) The boundary of f. The circular lists defining the boundary of f are: $[15, 16, 17]$, $[33, 31, 32, 31]$, $[13, 12, 14, 12, 11, 10, 9, 4, 29, 20, 19, 18, 20, 4]$. (c) The facial cycles of f.

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 4.1 *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 $x \in V(G)$, $\sigma_{\Gamma_1}(x) = \sigma_{\Gamma_2}(x)$. 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, ..., C_k. Then, Γ_1 and Γ_2 have the same face boundaries if: (a) for each C_i, $i = 1, \ldots, k$, 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 $i, j = 1, \ldots, k$ 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 $C_x, f_1, C_1, f_2, \ldots, C_y$ 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 $V^{left}_{\Gamma_1}(\vec{C}) = V^{left}_{\Gamma_2}(\vec{C})$. □

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*.

4.3 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.

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 4.1 *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 4.2 *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 1 of Lemma 4.1 if and only if, for each node μ of $\mathcal{T}$, the corresponding embedding of* skel(μ) *is edge-compatible with $\mathcal{H}$.*

Proof: Obviously, if G has an embedding satisfying Condition 1 of Lemma 4.1, then the corresponding embedding of *skel*(μ) 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 4.1. 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 *skel*(μ). Since the embedding of *skel*(μ) is assumed to be edge-compatible with $\mathcal{H}$, the triple (e', f', g') is embedded in clockwise order in *skel*(μ), 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 *skel*(μ) or the virtual edges whose pertinent graphs contain the edges of $\vec{C}$ form a simple cycle in *skel*(μ). Such a cycle in *skel*(μ) inherits the orientation of $\vec{C}$ in a natural way.

Definition 4.2 *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* skel(μ)*, all the vertices of* skel(μ) *that belong to $V_{\mathcal{H}}^{left}(\vec{C})$ and all the virtual edges that contain vertices of $V_{\mathcal{H}}^{left}(\vec{C})$ (except for the virtual edges of $\vec{C}'$ itself) are embedded to the left of $\vec{C}'$; and analogously for $V_{\mathcal{H}}^{right}(\vec{C})$.*

Lemma 4.3 *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 4.1 if and only if, for each node μ of $\mathcal{T}$, the corresponding embedding of* skel(μ) *is cycle-compatible with $\mathcal{H}$.*

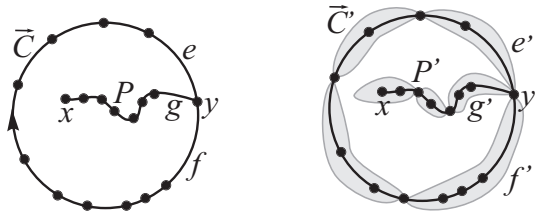
Proof: Obviously, if $\mathcal{G}$ is an embedding of G that satisfies Condition 2 of Lemma 4.1, then the corresponding embedding of *skel*(μ) is cycle-compatible with $\mathcal{H}$, for each node μ of $\mathcal{T}$.

To prove the converse, assume that *skel*(μ) 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. 4.3. 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 connecting x to a vertex of $\vec{C}$. Path P 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.

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 *skel*(μ). Let $\vec{C}'$ be the projection of $\vec{C}$ into *skel*(μ) (in other words, $\vec{C}'$ is the subgraph of *skel*(μ) 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



(a) (b)

Figure 4.3: Illustration for the proof of Lemma 4.3. Grey regions represent virtual edges of the skeleton of a node of $\mathcal{T}$.

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 $skel(\mu)$ 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 $skel(\mu)$. In particular, the edge of $skel(\mu)$ containing x is to the right of $\vec{C}'$. Since the embedding of $skel(\mu)$ 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 4.1, as claimed. □

Definition 4.3 *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 4.2 and 4.3, we obtain the following result, characterizing the positive instances of PEP in which G is biconnected.

Theorem 4.1 *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 4.1 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 $skel(\mu)$. We say that a virtual edge e of $skel(\mu)$ 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.

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. 4.4). 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.

(a) (b) (c)

Figure 4.4: Three examples of PEP instances $(G, H, \mathcal{H})$ with no embedding extension, even if each block of G admits an embedding extending the corresponding sub-embedding of $\mathcal{H}$. Black edges and vertices represent $\mathcal{H}$, gray edges and vertices belong to G but not to H. Note that instance (a) fails to satisfy Condition 3 of Lemma 4.4, instance (b) fails to satisfy Condition 2 of Lemma 4.4, and instance (c) has a non-trivial non-local H-bridge.

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.

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. 4.5.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. 4.5.b). In this case, we define f_K as an arbitrary face of $\mathcal{H}$.

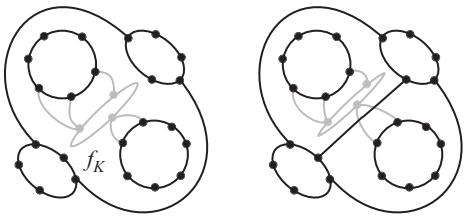


Figure 4.5: A non-local bridge is either necessarily contained in a face f_K (a) or causes a non-planarity (b).

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 4.4 *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 4.4 *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 $V_{\mathcal{H}}^{left}(\vec{C})$, i.e., they are 'inside' f'. To see that the claim holds, assume first that H' has no edge incident to x (see Fig. 4.6.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.

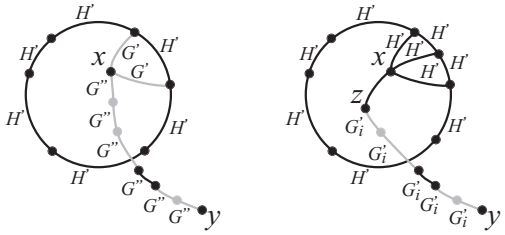


Figure 4.6: Illustration for the proof of Lemma 4.4. (a) H' has no edge incident to x. (b) H' has an edge incident to x.

Next, assume that H' has an edge incident to x (see Fig. 4.6.b). By the construction of G_1, each connected component of $G - x$ has at least one vertex adjacent to x in H. Moreover, the edges of $\mathcal{H}'$ incident to x form an interval in $\sigma_{\mathcal{H}}(x)$. Hence, $\mathcal{H}'$ 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''. Note that any two vertices

in $H' - x$ are inside the same face f'' of $\mathcal{H}''$ in $\mathcal{H}$ by condition 3 of the lemma, as 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'', 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. □

Observe that the second and third conditions of Lemma 4.4 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 4.5 *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. 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 4.5 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 chapter, we describe a more sophisticated algorithm that solves PEP in linear time.

4.4 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 4.4, uses the algorithms presented in Subsections 4.4 and 4.4 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.

G biconnected, H biconnected

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

Lemma 4.6 *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 4.1. Then, $\mathcal{G}$ satisfies Condition 2 of Lemma 4.1.*

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 4.1. 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 4.1. □

Due to Lemma 4.6, testing whether a planar embedding $\mathcal{G}$ exists satisfying Conditions 1 and 2 of Lemma 4.1 is equivalent to testing whether a planar embedding $\mathcal{G}$ exists satisfying Condition 1 of Lemma 4.1. Due to Lemma 4.2, 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 rooted 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 *skel*(μ) 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 *pertinent*(μ) 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 *pertinent*(μ). 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 *skel*(μ) is a cycle, the only

planar embedding of *skel*(μ) 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 *skel*(μ) 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 *skel*(μ) restrict the circular list of its incident virtual edges to the virtual edges $e_1, \ldots, e_h$ that contain an edge of H incident to x. Check if $l_{x(e_i)}$ precedes $f_{x(e_{i+1})}$ (for $i = 1, \ldots, h$, 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 *skel*(μ). 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 *skel*(μ) is a counterclockwise order of its virtual edges around $u(\mu)$. We describe how to verify if an embedding of *skel*(μ) 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 *skel*(μ) 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 *skel*(μ) 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 *skel*(μ) not in L_u and not in L_v, thus obtaining an embedding of *skel*(μ) 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 4.2 *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 *skel*(μ) 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 *skel*(μ) in $O(k_\mu)$ time. For each of such embeddings, Algorithm BB processes each node x of *skel*(μ) 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 all the nodes of *skel*(μ) results in a total $O(k_\mu)$ time, as every edge is incident to two nodes and the total number of edges in *skel*(μ) is $O(k_\mu)$.

If μ is a P-node, extracting the virtual edges of *skel*(μ) 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 $O(n)$. □

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 $(G(f), H(f), \mathcal{H}(f))$ and 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, 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 *skel*(μ) 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 *pertinent*(μ), and that is part of a simple cycle $\vec{C}$ of $H(f)$ not entirely contained in *pertinent*(μ); 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 4.7 *Consider any node μ of $\mathcal{T}(f)$. Then, an embedding of* skel(μ) *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 skel(μ)*, no vertex and no edge of* skel(μ) *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 4.3, if the edges of $\vec{C}$ determine a cycle $\vec{C}'$ of virtual edges of *skel*(μ), all the vertices of *skel*(μ) that are not in $\vec{C}$ and all the virtual edges of *skel*(μ) 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 $\mathcal{H}(f)$. □

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

Lemma 4.8 *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 $1 \leq i \leq k$ if and only if $p(\mu_j) =$* TRUE *for all $1 \leq j \leq k$.*

Proof: If $p(\mu_j) =$ TRUE for all $1 \leq j \leq k$, then $p(\mu_i) =$ TRUE. If $p(\mu_i) =$ TRUE for some $1 \leq i \leq k$, there exists a traversing path of μ_i that is part of a simple cycle $\vec{C}$ of $H(f)$ not entirely contained in *pertinent*(μ_i); however, as μ is an S-node, $\vec{C}$ does not entirely lie inside *pertinent*(μ), as otherwise it would lie inside *pertinent*(μ_i). Then, $\vec{C}$ consists of a traversing path of *pertinent*(μ_j), for all $1 \leq j \leq k$, and of a traversing path of the virtual edge of *skel*(μ) corresponding to the parent of μ in $\mathcal{T}(f)$, thus proving the lemma. □

Lemma 4.9 *Let μ be an R-node of $\mathcal{T}(f)$. If an edge e of* skel(μ) *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* skel(μ). *An embedding of* skel(μ) *is cycle-compatible with* $\mathcal{H}(f)$ *if and only if, for each face* g *of the embedding of* skel(μ), *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 *skel*(μ) is cycle-compatible with $\mathcal{H}(f)$. Let g be a face of the embedding with an edge e on its boundary 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 4.7, $\vec{C}$ projects to a directed cycle $\vec{C}'$ in *skel*(μ), and no vertex or edge of *skel*(μ) 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 *skel*(μ), every face satisfies condition (i) or condition (ii). We claim that the embedding of *skel*(μ) is cycle-compatible with $\mathcal{H}(f)$. To prove it, we use Lemma 4.7. Let $\vec{C}$ be a facial cycle of $\mathcal{H}(f)$ that projects to a simple cycle $\vec{C}'$ in *skel*(μ). Let e be any edge of $\vec{C}'$ and let g be the face to the right of e in the embedding of *skel*(μ). 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 *skel*(μ) is embedded to the right of $\vec{C}'$. By Lemma 4.7, the embedding of *skel*(μ) is cycle-compatible with $\mathcal{H}(f)$. □

Lemma 4.10 *Let* μ *be a P-node of* $\mathcal{T}(f)$. *There exist either zero or two virtual edges of* skel(μ) *containing a traversing path.*

Proof: If there exists one virtual edge e_i of *skel*(μ) containing a traversing path that is part of a simple cycle $\vec{C}$ of $H(f)$ not entirely contained in *pertinent*(e_i), another virtual edge of *skel*(μ) 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 *skel*(μ) 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 *skel*(μ) is a cycle, the only planar embedding of *skel*(μ) 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 *skel*(μ), 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 do this, check if Lemma 4.7 is satisfied, by first determining if the virtual edge e_p of $skel(\mu)$ representing the parent of μ in $\mathcal{T}(f)$ contains a traversing path P_p and, in case it does, by determining its orientation. By definition, P_p exists if and only if there exists a traversing path in $pertinent(\mu)$. Restrict $skel(\mu)$ to those edges $e_i \neq e_p$ with $p(e_i) = \text{TRUE}$ and denote by $skel'(\mu)$ the obtained graph. Check if the degree of $u(\mu)$ and $v(\mu)$ in $skel'(\mu)$ is even or odd. In the former case, P_p does not exist; set $p(\mu) = \text{FALSE}$ and $p(e_p) = \text{FALSE}$. In the latter case, P_p exists; set $p(\mu) = \text{TRUE}$ and $p(e_p) = \text{TRUE}$; the orientation of P_p is the only one that makes the number of edges e_i incident to $u(\mu)$ with $uv(e_i) = \text{TRUE}$ equal to the number of edges e_i incident to $u(\mu)$ with $uv(e_i) = \text{FALSE}$; 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 $skel(\mu)$. Consider every face g of $skel(\mu)$ 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 $p(e_j) = \text{FALSE}$, or $p(e_j) = \text{TRUE}$ and $uv(e_j) = \text{FALSE}$, for all edges e_j incident to g, and check whether $p(e_j) = \text{TRUE}$ and $uv(e_j) = \text{TRUE}$, for all edges e_j incident to g. If one of the two checks succeeds, the face does not violate Lemma 4.7, otherwise it does.

If μ is a P-node, check if an embedding of $skel(\mu)$ exists that is compatible with $\mathcal{H}(f)$ as follows. By Lemma 4.10, there exist either zero or two virtual edges of $skel(\mu)$ containing a traversing path. Then, consider the children μ_i of μ such that $p(\mu_i) = \text{TRUE}$. If zero or two such children exist, then the edge of $skel(\mu)$ corresponding to the parent ν of μ in $\mathcal{T}$ has no traversing path; if one such a child exists, then the edge of $skel(\mu)$ corresponding to ν has a traversing path. Denote by e_i and e_j the edges of $skel(\mu)$ containing a traversing path, if such edges exist, where possibly e_j corresponds to ν (in this case, set $p(e_j) = \text{TRUE}$, and set $uv(e_j) = \text{TRUE}$ if $uv(e_i) = \text{FALSE}$ and $uv(e_j) = \text{FALSE}$ otherwise). If there exists no edge e_i of $skel(\mu)$ such that $p(e_i) = \text{TRUE}$, then construct an embedding of $skel(\mu)$ 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 $skel(\mu)$, 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 $p(\mu) = \text{FALSE}$. If there exist two edges e_i and e_j such that $p(e_i) = \text{TRUE}$, $p(e_j) = \text{TRUE}$, and $p(e_l) = \text{FALSE}$ for every edge $e_l \neq e_i, e_j$, suppose that $uv(e_i) = \text{TRUE}$ and $uv(e_j) = \text{FALSE}$, the case in which $uv(e_i) = \text{FALSE}$ and $uv(e_j) = \text{TRUE}$ being analogous. Then, by Lemma 4.7, 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 $skel(\mu)$ as in Algorithm BB, except for the fact that the edges of $skel(\mu)$ 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 4.3 *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 $\mu_1, \ldots, \mu_{k_\mu}$ 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 $skel(\mu)$ 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 (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 checking the flags $p(\mu_i)$ and $uv(\mu_i)$, for $i = 1, \ldots, k$.

If μ is an R-node, the construction of $skel'(\mu)$ can be done in $O(k_\mu)$ time, as such a graph can be obtained from $skel(\mu)$ by simply checking flag $p(e_i)$, for each edge e_i in $skel(\mu)$. Then, the degree of $u(\mu)$ and $v(\mu)$ in $skel'(\mu)$, 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 $skel(\mu)$ and the number of edges in $skel(\mu)$ is $O(k_\mu)$, the total time spent for the test on the faces of $skel(\mu)$ is $O(k_\mu)$.

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

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 4.4. Let f be a face of $\mathcal{H}'$. Let $V(f)$ be the set of vertices of H' incident to f. Let $H(f)$ be the subgraph of H' induced by $V(f)$. 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. Note that H^+ has the same vertex set as H'. Let $G(f)$ be the subgraph of H^+ induced by $V(f)$. Any embedding of H^+ extending $\mathcal{H}$ also extends $\mathcal{H}'$, and vice versa. Also, in any embedding of H^+ extending $\mathcal{H}$ the edges of $G(f)$ not belonging to $H(f)$ are embedded inside f.

Lemma 4.11 *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 4.12 *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 4.1, the embeddings of $G(f)$ extending $\mathcal{H}(f)$ are obtained by specifying a compatible embedding for each skeleton of $\mathcal{T}$. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be 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 show that they are also mutually visible with respect to $\mathcal{G}_2$. In view of Theorem 4.1, 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 4.7, 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.

We prove 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 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 $K_1, \ldots K_s$ be the local H-bridges assigned to f and such that all their attachments are on the boundary of f'. 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 $K_1, \ldots K_s$ can be embedded inside f'.

First, observe that the boundary of f' is a simple cycle C', as H^+ is biconnected. Also, 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 $K_1, \ldots, K_s$ can be embedded inside C', proceed by induction on s. If $s = 1$ the statement is true. Assume that $s \geq 2$ and that bridge K_1 has been successfully embedded into f'. The embedding of K_1 partitions f' into subfaces $f'_1, \ldots, f'_t$, each one bounded by a simple cycle, as 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. If 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, K_1 and K_i have a four-vertex conflict of on $\vec{C}'$, contradicting the second condition of the claim. If 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, K_1 and K_i have a four-vertex conflict on $\vec{C}'$, again contradicting the second condition of the claim. Thus, we can assign to each K_i a subface f'_j containing all its attachments. By induction, all the K_i's can be embedded into their assigned faces, thus proving the claim.

The proof that condition *(b)* holds 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 Claim 2, since it can be extended to $\mathcal{G}$. Then, every embedding of H^+ that extends $\mathcal{H}$ satisfies the two conditions of Claim 2: For the first condition, it is a consequence of Claim 1, and for the second condition, it is obvious. We conclude that every embedding of H^+ extending $\mathcal{H}$ 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 4.13 *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 $a_1, a_2, \ldots, a_h$ 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 [KK03]. 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 $a_3, \ldots, a_h$ of K. Again, each such a check is performed in constant time [KK03]. 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 $a_3, \ldots, a_h$, 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 [KK03] 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 4.4 *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 4.12.

By Lemma 4.13, 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 4.3, 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 4.2, Algorithm BB runs in $O(n)$ time, hence the total running time of Algorithm BA is $O(n)$. □

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 4.4 can be checked in linear time. The first condition can be checked in linear time by Lemma 4.13. The second and the third conditions can be checked in linear time by the following two lemmata.

Lemma 4.14 *Let $(G, H, \mathcal{H})$ be an instance of* PEP*, where G is connected. Let $G_1, \ldots, G_t$ 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 $H_1, \ldots, H_t$ alternate around a vertex of $\mathcal{H}$.*

Proof: Assume that every edge e of H has an associated label indicating the block of G that contains e. Also, associate to each block two integer counters that will be used in the algorithm.

We 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. We call *x-edge* any edge of H incident to x and *x-block* any block of G that contains at least one x-edge. 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 number in the counter associated with G_i, by considering 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 visited so far. An x-block is *active* if some but not all of its x-edges have already been visited. A stack containing the active x-blocks is maintained. At the beginning all the counters are set to zero and the stack is empty.

For every visited edge e, TEST(x) performs the following steps:

1. Increment the counter of visited x-edges of the block G_i containing e.
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 $\mathcal{H}$.

We prove the correctness of TEST(x). 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 e is not the first visited edge of G_i. Hence, 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. Thus, H_i and H_j 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 4.15 *Let $(G, H, \mathcal{H})$ be an instance of* Pep *where G is connected. Let $G_1, \ldots, G_t$ 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 $H_1, \ldots, H_t$ 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 show that the existence of a PC-obstruction can be tested in linear time. Note that it is sufficient to test the existence of a minimal PC-obstruction.

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. Note 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 to determine the existence of a minimal PC-obstruction. The algorithm tests all the edges of $E(G) \setminus E(H)$ one by one. For any 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 4.13. 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 4.13. Such a face, 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 4.4, 4.13, 4.14 and 4.15 with Theorem 4.4, we obtain the following result.

Theorem 4.5 PEP *can be solved in linear time when restricted to instances* $(G, H, \mathcal{H})$ *where* G *is connected.*

Proof: By Lemma 4.13, 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 4.13. 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 4.4. 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 4.4, 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 alternating around a vertex of $\mathcal{H}$. If this is the case, we reject the instance, otherwise we continue. This step can be implemented in linear time, due to Lemma 4.14.

Finally, we check the existence of PC-obstructions, which by Lemma 4.15 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 4.4. □

Next, we deal with the instances $(G, H, \mathcal{H})$ of PEP in which G is disconnected and H arbitrary. We use Lemma 4.5 directly, and show that the two conditions of the lemma can be checked in linear time. The first condition of Lemma 4.5 can be checked in linear time by Theorem 4.5. 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 4.6 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 4.5, $(G, H, \mathcal{H})$ 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 4.5, 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 $\mathcal{H}_j$ has at least two outer faces.

We now describe the algorithm that tests the second condition of Lemma 4.5. 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.

4.5 Generalizations of Pep

Several generalizations of the Partially Embedded Planarity problem naturally arise. The following two generalizations are readily seen to be NP-complete, since they are special cases of Crossing number and Maximum planar subgraph, respectively:

- Deciding whether an embedding $\mathcal{H}$ can be extended to a planar drawing of G with at most k crossings; and
- deciding whether 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:

Minimum Rerouting Partially Embedded Planarity Given a planar graph G, a planar embedding $\mathcal{H}$ of a subgraph H of G, and an integer $k \geq 0$, decide whether G admits a planar embedding $\mathcal{G}$ in which at least k edges of H are embedded the same as in $\mathcal{H}$; and

Maximum Preserved Partially Embedded Planarity Given a planar graph G, a planar embedding $\mathcal{H}$ of a subgraph H of G, and an integer $k \geq 0$, decide 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 $\mathcal{H} \setminus F$.

In the following theorem we prove that even these two problems are NP-hard.

Theorem 4.7 Minimum Rerouting Partially Embedded Planarity *and* Maximum Preserved Partially Embedded Planarity *are NP-hard.*

Proof: The proof is obtained by means of a reduction from SteinerTree in planar graphs, which is known to be NP-hard [GJ77]. Let $G = (V, E)$ be a planar graph and let $T \subseteq V$ be a set of terminals. Choose an embedding $\mathcal{G}$ of G and let H be the dual of G with embedding $\mathcal{H}$. For each terminal $t \in T$ we add a new vertex v_t to H and prescribe it inside the face $t^\star$ that is dual to t. Moreover let S be the edge set of any connected graph on the vertices v_t. We set $G' := H + S$.

Now consider the problem of identifying a set F of edges of H such that $G - F$ can be drawn planar and such that the subgraph $H - F$ has embedding $\mathcal{H} - F$. Clearly S can be drawn if and only if we remove edges of H such that all vertices v_t lie in the same face. This is equivalent to the property that the set $F^\star$ of edges dual to F is a Steiner Tree in G with terminal set T.

This shows that Maximum Preserved Partially Embedded Planarity is NP-hard. Moreover, as the vertices v_t form a separate connected component, we can reinsert the edges of F without crossings into the drawing, i.e., it is sufficient to reroute the edges in F. This shows that Minimum Rerouting Partially Embedded Planarity is NP-hard, as well. □

In the case of Maximum Preserved Partially Embedded Planarity, the NP-hardness result holds even if H is connected. Namely, connect each vertex v_t to an arbitrary vertex of its prescribed face and choose for S the edge set of a star graph on the vertices v_t. The same result does not hold for Minimum Rerouting Partially Embedded Planarity, as in that case the reduction relies on the property that every edge of each face can be removed and reinserted after drawing S, that is not the case if H is connected. We leave open the question whether Minimum Rerouting Partially Embedded Planarity is NP-hard if the prescribed graph H is connected.

4.6 Simultaneous Embedding with Fixed Edges

The results presented in this chapter solve special cases of the so called simultaneous embedding problem.

A Simultaneous Embedding with Fixed Edges (in the following called SEFE, for short) of a tuple $(G_1 = (V, E_1), G_2 = (V, E_2), \ldots, G_k = (V, E_k))$ of graphs on the same set of n vertices is a tuple $(\Gamma_1, \Gamma_2, \ldots, \Gamma_k)$ of drawings such that: (i) Γ_i is a planar drawing of G_i, for each $i = 1, 2, \ldots, k$; (ii) each vertex $v \in V$ is drawn on the same point in Γ_i, for every $i = 1, 2, \ldots, k$; (iii) each edge $(u, v) \in E_i \cap E_j$ is represented by the same Jordan curve in Γ_i and in Γ_j.

Several combinatorial results are known on the problem of determining which graphs always have a SEFE: Erten and Kobourov proved that a tree and a path always have a SEFE with at most one bend per edge [EK04]. Di Giacomo and Liotta proved that an outerplanar graph and a cycle always have a SEFE with one bend per edge [DL07]. Frati proved that a planar graph and a tree always have a SEFE, while there exist pairs of outerplanar graphs that have no SEFE [Fra06]. Fowler *et al.* characterized the class of planar graphs that always have a SEFE with any other planar graph [FJKS08].

Further, the problem of testing whether a given set of planar graphs have a SEFE has also been deeply investigated. Gassner *et al.* proved that deciding whether three graphs have a SEFE is an NP-complete problem [GJP+06]. Fowler *et al.* proved that testing whether two biconnected outerplanar graphs have a SEFE and that testing whether two planar graphs whose union is homeomorphic to K_5 or to $K_{3,3}$ have a SEFE are polynomial-time-solvable problems [FJKS08]. Fowler *et al.* proved that whether two planar graphs have a SEFE is a polynomial-time-solvable problem if one of the graphs has at most one cycle [FGJ+08]. Estrella-Balderrama *et al.* proved that deciding whether two graphs have a SEFE is an NP-hard problem if the further constraint that the edges have to be represented by straight-line segments is considered [EBGJ+07] (in such a setting the problem is usually regarded as *geometric simultaneous embedding*). Determining the time complexity of testing whether two planar graphs have a SEFE is, in general, an open problem.

As a consequence of the results we presented on the PEP problem, deciding whether two graphs have a SEFE is a linear-time solvable problem if one the graphs has a fixed embedding. Namely, let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two planar graphs and let $\mathcal{G}_2$ be a planar embedding of G_2. Denote by $G_1 \cap G_2$ a planar graph whose vertex set is V and whose edges are those belonging to both E_1 and E_2. Denote by $\mathcal{G}_{1\cap 2}$ the planar embedding of $G_1 \cap G_2$ obtained as $\mathcal{G}_2$ restricted to the edges of $G_1 \cap G_2$.

Theorem 4.8 *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.

Proof: Suppose that G_1 and G_2 have a SEFE (Γ_1, Γ_2) in which the planar embedding of G_2 is $\mathcal{G}_2$. We prove that $(G_1, G_1 \cap G_2, \mathcal{G}_{1\cap 2})$ is a positive instance of PEP. Denote by $\mathcal{G}_1$ the planar embedding of G_1 corresponding to Γ_1. We claim that $\mathcal{G}_1$ restricted to $G_1 \cap G_2$ is $\mathcal{G}_{1\cap 2}$. However, this is easily proved by observing that, if the embedding of $G_1 \cap G_2$ is not $\mathcal{G}_{1\cap 2}$, then the embedding of G_2 corresponding to Γ_2 is not $\mathcal{G}_2$, a contradiction.

Suppose that $(G_1, G_1 \cap G_2, \mathcal{G}_{1\cap 2})$ is a positive instance of PEP. Then, there exists a planar embedding $\mathcal{G}_1$ of G_1 such that $\mathcal{G}_1$ restricted to $G_1 \cap G_2$ is $\mathcal{G}_{1\cap 2}$. We show how to construct a SEFE (Γ_1, Γ_2) of (G_1, G_2). Drawing Γ_1 is any planar drawing with embedding $\mathcal{G}_1$. Consider the planar drawing $\Gamma_{1\cap 2}$ of $G_1 \cap G_2$ that is obtained from Γ_1 by removing the edges not in $G_1 \cap G_2$. Observe that $\Gamma_{1\cap 2}$ corresponds to embedding $\mathcal{G}_{1\cap 2}$, by definition of PEP. Since

$\mathcal{G}_2$ restricted to $G_1 \cap G_2$ is also $\mathcal{G}_{1\cap 2}$, drawing $\Gamma_{1\cap 2}$ can be completed to a planar drawing Γ_2 of G_2, thus obtaining a SEFE (Γ_1, Γ_2) of (G_1, G_2). □

4.7 Conclusions

In this chapter 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. Also, we proved that some generalizations of Partially Embedded Planarity where the minimum number of crossings, of edges to be rerouted, or of edges to be deleted are requested, are NP-hard. Finally, we proved that Partially Embedded Planarity is the same problem as the Simultaneous Embedding with Fixed Edges problem, when one of the two graphs has a fixed embedding, which implies that such a problem can be solved in linear time, as well.

One direction of future work is to extend the algorithm such that it not only rejects unsolvable Pep instances but also provides a small certificate proving that the extension is not possible. To this aim, an intriguing goal would be to have a characterization of solvable Pep instances via an obstruction set. The algorithm should then be modified to find an obstruction quickly in case an instance is not solvable.

Another interesting question is whether our approach can be generalized to work for embeddings on the torus or on surfaces of higher genus.

Chapter 5

Minimum-Depth Embeddings

In this chapter[1] we deal with a problem concerning the optimization of quality measures of embeddings. Namely, we consider the problem of efficiently computing an embedding of a planar graph that is optimal with respect to some topological distance measures. In particular, we aim at minimizing the maximum distance of the internal faces of an embedding of a given planar graph from the external face. Among the several possible ways to define the distance between two faces, we consider the one, called *depth*, that is based on the adjacency relationship in which two faces are adjacent if they share an edge.

Problem 5.1 (Minimum-Depth Embedding) *Given a planar graph G, what is the time complexity of computing an embedding of G with minimum depth?*

This problem is worth of study for several reasons, as many researchers pointed out that the quality of a planar embedding can be measured in terms of its depth and as the computation of an optimal embedding with respect to such a measure is the basic step of many drawing algorithms.

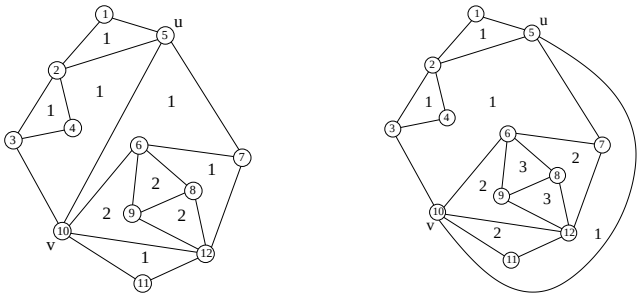
In this chapter we present an $O(n^4)$-time algorithm for computing a minimum-depth embedding of any given planar graph. This bound improves on the best previous bound [BM90] by an $O(n \log n)$ factor. As a side effect, our algorithm improves the bounds of several algorithms that require the computation of a minimum-depth embedding.

[1] Some of the contents of this chapter are a joint work with Giuseppe Di Battista and Maurizio Patrignani, appeared in [ADP07] and to appear in [ADP10].

5.1 Introduction

As pointed out in [BM90, PT00, Piz05], the quality of a planar embedding of a planar graph can be measured in terms of the maximum distance of its vertices from the external face. Such a distance can be given in terms of different incidence or adjacency relationships between vertices and faces. For example, if we consider an adjacency relationship in which two faces are adjacent when they share a vertex, then the maximum distance to the external face is called *radius* [RS84]. If the adjacency relationship is such that two vertices are adjacent when they are endpoints of an edge, then the maximum distance to the external face is called *width* [DLT84]. If the adjacency relationship is such that two vertices are adjacent when they are on the same face and the external face is adjacent to all its vertices, then the maximum distance to the external face is called *outerplanarity* [Bak94]. Finally, if we consider two faces to be adjacent when they share an edge, then the maximum distance to the external face is called *depth* [BM88]. Fig. 5.1 shows two embeddings of the same graph where internal faces are labeled with their depth.

Pizzonia and Tamassia present an algorithm to minimize the maximum distance of the biconnected components of the graph from the external face, where two biconnected components are adjacent if they share a cut-vertex [PT00]. This measure, which they also call "depth", is a rougher indicator of the quality of the embedding but can be computed in $O(n)$ time.



(a) (b)

Figure 5.1: Two embeddings of the same graph G lead to different values of depth. Internal faces are labeled with their distance from the external face.

The algorithms that compute a planar embedding to minimize the maxi-

mum distance of any internal vertex or face from the external face, for some definition of distance, have several applications. Let us give a few examples. The algorithm by Dolev, Leighton, and Trickey for drawing planar graphs with asymptotically optimal area [DLT84] requires the computation of the embedding with minimum width. In [Bak94] Baker gives approximation algorithms on planar graphs for many NP-complete problems, including maximum independent set and minimum vertex cover. The time complexity and the optimality bounds of such algorithms depend on the outerplanarity of the graph. Finally, the algorithm by Di Giacomo et al. [DDLM05] for constructing minimum radial drawings of planar graphs relies on the computation of their outerplanarity.

In [BM90], Bienstock and Monma present an algorithm to compute a planar embedding of an n-vertex planar graph with minimum maximum distance to the external face in $O(n^5 \log n)$ time. The distance they consider is the depth. However, they observe that it is possible to compute radius, width, and outerplanarity by modifying and simplifying the algorithm for the minimum depth, since such distance measures are intrinsically simpler to compute than the depth. In [Kam07], Kammer focuses on such simpler distance measures and improves the complexity bounds for computing them by presenting an algorithm that computes the outerplanarity of an n-vertex planar graph in $O(n^2)$ time. Also, he observes that simple variations of his algorithm can lead to compute the radius in $O(n^2)$ time and the width in $O(n^3)$ time.

The algorithm of Bienstock and Monma for computing the minimum-depth embedding is based on the decomposition of the graph into its biconnected and triconnected components. The general approach is the one of selecting a positive integer k and checking if an embedding exists with depth k. A binary search is done to determine the optimal value of k. For each selected k the decomposition of G is visited associating to each component μ a left and a right weight, corresponding to the distances of the left and right border of μ from the external face of G. Such weights are independent on the embedding of the other components. The components are then visited to check if their weights can be composed to construct an embedding with depth k. The space complexity of the algorithm is not analyzed in the paper.

In this chapter we present an algorithm that improves the time bound of [BM90] to $O(n^4)$ time. As a side effect, we improve the time bound of the algorithms that need to compute a planar embedding with minimum depth.

Our approach is inspired by the methods in [BM90] and develops on top of such methods several new techniques. As in [BM90], we decompose the graph into bi- and tri-connected components, using BC-trees and SPQR-trees [DT96b]. However, we are able to solve the problem on each biconnected component,

with a given edge on the external face, in $O(n^3)$ time. Then, we use techniques analogous to those in [BM90] for assembling the results on each biconnected component into a general solution. Among the techniques presented in this chapter, a key issue, that might have other applications, is the ability of representing implicitly and with reasonable size all the possible values of depth of each triconnected component. The space complexity of the algorithm is $O(n^3)$.

The chapter is organized as follows. Section 5.2 provides an informal high-level description of the algorithm and a hint of its computational complexity. Section 5.3 gives basic definitions. Section 5.4 deals with the combinatorial structure of the depth of the planar embeddings and develops a theory of the set of integer pairs that is exploited in the algorithm. Section 5.5 presents the algorithm for biconnected graphs and Section 5.6 extends the algorithm to general connected graphs. In Section 5.7 we give concluding remarks and we further compare our approach with the one in [BM90].

5.2 High-Level Description of the Algorithm

In this section we give a high-level informal description of the algorithm for computing the minimum value of the depth among all the embeddings of a planar graph. The details of the algorithm are the subject of the next sections, which also describe how to actually obtain a minimum-depth embedding of the graph. First, we show how to handle biconnected graphs. Second, we show how to extend the algorithm to simply connected graphs.

The crucial ingredient of the algorithm for biconnected graphs is focusing on all the possible embeddings of the subgraphs that are attached to G by a separation pair. By using the SPQR-tree decomposition, we decorate each separation pair of G with suitable labels that describe the properties of all the embeddings of its associated subgraph with respect to the depth of the inner faces. In particular, we associate to each separation pair an infinite set of integer pairs. Each pair $\langle x, y\rangle$ represents the fact that the subgraph associated to the separation pair admits an embedding, with the separation pair on the external face, such that each internal face has depth at most x (y) with respect to the "left border" ("right border") of the subgraph.

Notice that, being able to compute such set of pairs for the root of the hierarchy allows us to find the value of a minimum-depth embedding when the starting edge is on the external face. By choosing all the possible starting edges and comparing the obtained values the problem is solved.

However, this strategy requires to represent in a succinct way the infinite

sets of integer pairs and to efficiently compute these succinct representations.

For solving the first problem we study in Section 5.3 the properties of such sets, showing that a finite subset of pairs, that we call "gist" and whose size is linear in the size of the subgraph associated to the separation pair, can be used to implicitly represent the whole infinite set.

For the second problem, the one of efficiently compute succinct representations of infinite sets, we compute gists bottom-up on the SPQR-tree. This is the most difficult part, especially for parallel nodes, as the factorial number of possible permutations must be taken into account. This is also the most computationally expensive step, requiring an overall $O(n^3)$ time to compute the gists of all the parallel nodes and yielding an $O(n^4)$ time algorithm when the root of the decomposition is chosen in all possible ways.

The algorithm for simply connected graphs starts from a decomposition of the input graph into its biconnected components and follows an approach similar to the one used in [BM90]. The basic step of this algorithm is a modified version of the $O(n^3)$-time algorithm to compute the minimum depth of all the embeddings of a biconnected graph with a specified edge on the external face. Such a modified version takes into account the fact that some specific vertices may have a subgraph attached to them. This algorithm is launched at most twice for each edge of each biconnected component, hence yielding an $O(n^4)$ time algorithm for the whole graph.

5.3 Properties of Sets of Integer Pairs

In this section we give basic properties of pairs $\langle x, y\rangle$ of non-negative integers. We say that pair $p_1 = \langle x_1, y_1\rangle$ *precedes with respect to* x (*precedes with respect to* y) pair $p_2 = \langle x_2, y_2\rangle$ when $x_1 \leq x_2$ ($y_1 \leq y_2$). We denote this relationship by $p_1 \preceq_x p_2$ ($p_1 \preceq_y p_2$). For example $\langle 3, 2\rangle \preceq_y \langle 4, 3\rangle$. We say that pair p_1 *precedes* pair p_2 when $p_1 \preceq_x p_2$ and $p_1 \preceq_y p_2$. We denote this relationship by $p_1 \preceq p_2$. For example $\langle 3, 2\rangle \preceq \langle 4, 3\rangle$. Since this relationship is reflexive, antisymmetric, and transitive, it determines a poset $(P, \preceq)$, where P is the set of non-negative integer pairs. We say that two pairs p_1 and p_2 are *incomparable* if none of them precedes the other, i.e., if $p_1 \not\preceq p_2$ and $p_2 \not\preceq p_1$, and we denote it by $p_1 \nsim p_2$. For example, $\langle 3, 2\rangle \nsim \langle 2, 3\rangle$.

A set $S \subseteq P$ of pairs of non-negative integers is *succinct* if the pairs of S are pairwise incomparable. In terms of poset theory, a succinct set is an independent subset of $(P, \preceq)$ or an antichain. Given two sets S and S' of pairs of integers, S' *precedes* S if for any pair $p \in S$ there exists at least one pair

$p' \in S'$ such that $p' \preceq p$. We write $S' \preceq S$ to mean that S' precedes S. For example $\{\langle 0,4\rangle\langle 3,3\rangle\langle 5,4\rangle\} \preceq \{\langle 0,5\rangle\langle 4,5\rangle\}$. Also, S' *reduces* S if $S' \preceq S$ and $S' \subseteq S$. Further, if S' is succinct and reduces S, then S' is a *gist* of S. In terms of poset theory, a *gist* of S is a set of minimal elements of $(S, \preceq)$. A pair $p \in S$ is a *minimal element* if there exists no pair $p' \in S$ such that $p' \preceq p$. For example, $\{\langle 0,4\rangle\langle 3,3\rangle\}$ is a gist of $\{\langle 0,4\rangle\langle 3,3\rangle\langle 5,4\rangle\}$.

The following property descends from the poset theory.

Property 5.1 *Let S be a possibly infinite set of non-negative integer pairs and let $p_1 = \langle x_1, y_1\rangle$, $p_2 = \langle x_2, y_2\rangle$, and $p_3 = \langle x_3, y_3\rangle$ be three elements of S.*

1. *The gist of S is unique and is the smallest set preceding S.*
2. *If S is succinct, then $p_1 \preceq_x p_2 \Leftrightarrow p_2 \preceq_y p_1$.*
3. *If S is succinct, then the relationship $\preceq_x$ induces a total order on S. Such a total order is an inverse total order with respect to relationship $\preceq_y$.*
4. *If S is finite, denote by $x^{\max}(S)$ ($y^{\max}(S)$) the maximum value of x_i (y_i) in any pair $\langle x_i, y_i\rangle \in S$. If S is succinct, then $|S| \leq 1 + x^{\max}(S)$ and $|S| \leq 1 + y^{\max}(S)$.*
5. *If $p_1 \preceq_x p_2 \preceq_x p_3$, then $p_1 \nsim p_2$, $p_2 \nsim p_3 \Rightarrow p_1 \nsim p_3$.*

In the following, the gist of S is denoted by $\hat{S}$.

5.4 The Combinatorial Structure of Planar Embeddings and their Depths

Let $\mathcal{T}$ be the SPQR-tree of a biconnected planar graph G, rooted at a given Q-node corresponding to a reference edge e. Let μ be a node of $\mathcal{T}$. Observe that any embedding Γ_G of G with e on the external face corresponds to an embedding Γ_{G_μ} of the pertinent graph G_μ of μ with the poles on the external face. Also, the external face of Γ_{G_μ} corresponds to two faces of Γ_G, which can be arbitrarily called *left* and *right external faces of* G_μ, and denoted by f_l^μ and f_r^μ. For example, Fig. 5.2(a) shows a subtree rooted at an R-node R_1 of an SPQR-tree of a graph G and Fig. 5.2(b) shows an embedding of G. The embedding of G induces an embedding of the pertinent graph of R_1 (shown in grey), whose external face corresponds to the two faces labeled $f_l^{R_1}$ and $f_r^{R_1}$.

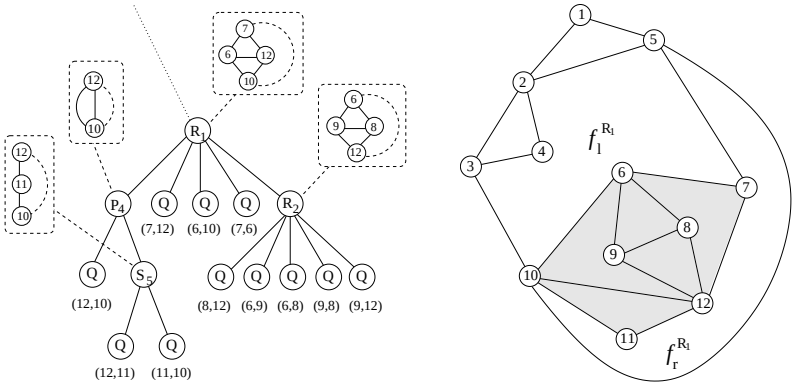
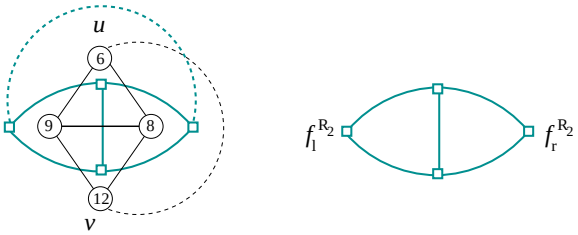


Figure 5.2: (a) The subtree rooted at a node R_1 of the SPQR-tree of a graph G. (b) An embedding of G where the pertinent graph of R_1 is drawn with gray faces and the two external faces $f_l^{R_1}$ and $f_r^{R_1}$ of R_1 are shown.

Following the approach of [BM90], we use a definition of faces f_l^μ and f_r^μ which is independent on the embedding Γ_G of G and only depends on Γ_{G_μ}. Let (u, v) be the virtual edge of μ that represents in μ the portion of G containing e and denote by G_μ^+ the graph obtained by adding edge (u, v) to the pertinent graph G_μ of μ. Suppose G_μ^+ is planarly embedded with edge (u, v) on the external face. The (u, v)-*dual* of G_μ is obtained by computing the dual of G_μ^+ and by then removing the edge of the dual corresponding to (u, v). The faces incident to the removed edge are f_l^μ and f_r^μ. Fig. 5.3 shows the construction of the (u, v)-dual graph of component R_2 of the SPQR-tree represented in Fig. 5.2(a).

A component μ *satisfies* the pair of non-negative integers $\langle x, y\rangle$ if its pertinent graph G_μ admits an embedding Γ_{G_μ}, with its poles on the external face, where it is possible to find a partition of the set of its internal faces into two sets, denoted by F_l and F_r, such that all faces in F_l have distance from f_l^μ less or equal than x and all faces in F_r have distance from f_r^μ less or equal than y. Fig. 5.4 shows an embedding of an S-node S_2 and three possible partitions of its faces corresponding to integer pairs $\langle 0, 2\rangle$, $\langle 1, 2\rangle$, and $\langle 2, 0\rangle$.

The set of integer pairs $\langle x, y\rangle$ satisfied by component μ is the *admissible set of* μ, and is denoted by $A(\mu)$. Observe that, as x and y are non-negative



(a) (b)

Figure 5.3: Construction of the (u, v)-dual graph of component R_2 of the SPQR-tree represented in Fig. 5.2(a). (a) Virtual edge (u, v) is drawn dashed, the pertinent graph of R_2 is drawn black, and the dual graph is drawn with thicker gray edges and square vertices. (b) The (u, v)-dual graph of R_2.

integers, Property 5.1 applies to admissible set $A(\mu)$. Also, observe that $p \in A(\mu)$ implies $p' \in A(\mu)$ for any p' such that $p \preceq p'$. We call this property *continuity*. Continuous sets of non-negative integer pairs such as $A(\mu)$ are infinite. In the following we show that $A(\mu)$, although infinite, can be efficiently represented and managed by means of its gist $\hat{A}(\mu)$, whose size is linear in the number of edges of G_μ.

Properties of Gists of Admissible Sets

With respect to the gist of a generic set of integer pairs, the gist of an admissible set satisfies additional properties that can be used to simplify the operations and to bound the complexity of the algorithms. Let μ be a component, G_μ its pertinent graph, and n_μ the number of vertices of G_μ.

Property 5.2 *If $\langle x, y\rangle \in \hat{A}(\mu)$, then $\langle y, x\rangle \in \hat{A}(\mu)$.*

Proof: Suppose, for a contradiction, that $\langle x, y\rangle \in \hat{A}(\mu)$ and $\langle y, x\rangle \notin \hat{A}(\mu)$. Since $\langle x, y\rangle \in A(\mu)$, flipping the embedding of the pertinent graph of μ which satisfies $\langle x, y\rangle$ yields $\langle y, x\rangle \in A(\mu)$. Hence, since $\langle y, x\rangle \notin \hat{A}(\mu)$, there exists a pair $\langle x', y'\rangle \in A(\mu)$ such that $\langle x', y'\rangle \preceq \langle y, x\rangle$. Hence, $\langle y', x'\rangle \in A(\mu)$. By construction, $\langle y', x'\rangle \preceq \langle x, y\rangle$, contradicting the hypothesis $\langle x, y\rangle \in \hat{A}(\mu)$. □

Property 5.3 *$\hat{A}(\mu)$ contains exactly one pair $\langle x, y\rangle$ with $x = 0$ and one pair $\langle x', y'\rangle$ with $y' = 0$. In such pairs, $y = x' = O(n_\mu)$.*

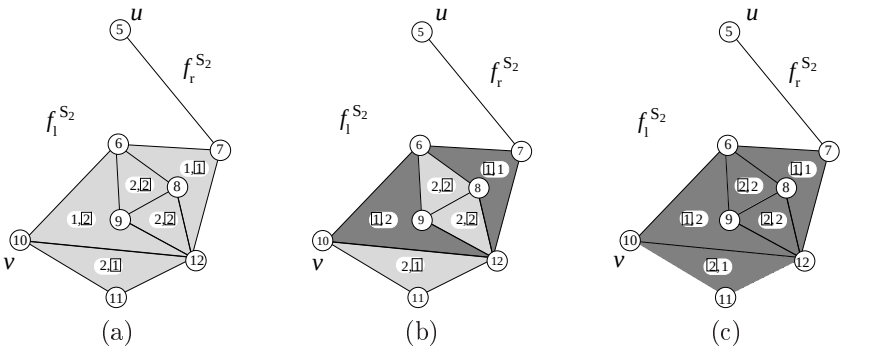


Figure 5.4: An embedding of the pertinent graph of an S-node S_2. Figures (a), (b), and (c) represent three possible partitions of the internal faces. Dark-shaded faces belong to F_l and light-shaded faces belong to F_r. Each face is labeled with two integers representing its distance from $f_l^{S_2}$ and $f_r^{S_2}$, respectively. Labels lying inside a square are those considered in the corresponding partition. The pictures show that S_2 satisfies pairs: $\langle 0, 2\rangle$ (a), $\langle 1, 2\rangle$ (b), and $\langle 2, 0\rangle$ (c).

Proof: Let F be the set of internal faces of an arbitrary embedding Γ_{G_μ} of G_μ. Consider the trivial partition of F into two sets such that one is empty and the other one coincides with F. Such a partition implies that $A(\mu)$ contains a pair $\langle x^*, y^*\rangle$ with $x^* = 0$ and $y^* \leq |F|$. Since any pair $\langle x, y\rangle$ preceding $\langle x^*, y^*\rangle$ has $x = 0$, $\hat{A}(\mu)$ contains at least one pair $\langle x, y\rangle$ with $x = 0$. Two such pairs can not be contained into $\hat{A}(\mu)$ since $\hat{A}(\mu)$ is succinct. The bound on the value of y is due to the fact that $y \leq y^* \leq |F|$ and that $|F| = O(n_\mu)$. Analogous considerations show that there exists exactly one pair $\langle x', y'\rangle \in \hat{A}(\mu)$ with $x' = O(n_\mu)$ and $y' = 0$. By Property 5.2 we have $y = x'$. □

Property 5.4 *Let $\langle x, y\rangle$ be a pair in $\hat{A}(\mu)$ satisfied by the embedding Γ_{G_μ}. There exists at least one face of Γ_{G_μ} that is at distance x from f_l^μ and at distance greater than y from f_r^μ.*

Proof: Consider the partition of the internal faces of Γ_{G_μ} into the two sets F_l and F_r that shows that Γ_{G_μ} satisfies $\langle x, y\rangle$. At least one face in F_l is at distance x from f_l^μ, otherwise Γ_{G_μ} would satisfy $\langle x-1, y\rangle$ and $\langle x, y\rangle$ would

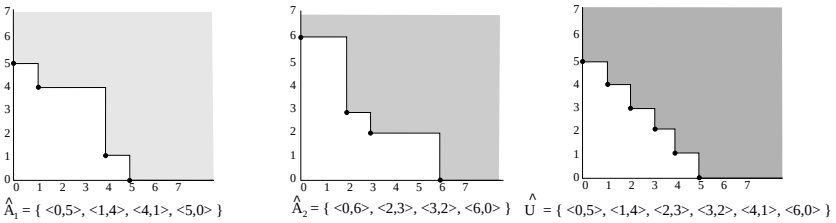


Figure 5.5: Computation of $\hat{U}$, where $U = A_1 \cup A_2$, given $\hat{A}_1$ and $\hat{A}_2$. An integer pair $\langle x, y \rangle$ belonging to a gist is represented by a black circle with coordinates (x, y).

not be in $\hat{A}(\mu)$. Denote by $F_{l,x}$ the set of faces of F_l that are at distance x from f_l^μ. If all faces in $F_{l,x}$ were at distance less or equal than y from f_r^μ, then the partition $F_l \setminus F_{l,x}$ and $F_r \cup F_{l,x}$ could be used to show that Γ_{G_μ} satisfies $\langle x-1, y \rangle$, contradicting the hypothesis that $\langle x, y \rangle$ is in $\hat{A}(\mu)$. □

Property 5.5 *$\hat{A}(\mu)$ is finite and $|\hat{A}(\mu)|$ is $O(n_\mu)$.*

Proof: By Property 5.3, $\hat{A}(\mu)$ contains two pairs $\langle x_1, y_1 \rangle$ and $\langle x_2, y_2 \rangle$, with $x_1 = 0$ and $x_2 = O(n_\mu)$. Since, by Property 5.1.3, $\hat{A}(\mu)$ is totally ordered with respect to the $\preceq_x$ relationship, it follows that $|\hat{A}(\mu)| = O(n_\mu)$. □

Some steps of the algorithm described in this chapter require the computation of unions and intersections among admissible sets. Since such sets are infinite, we perform these operations focusing on their finite representations. In the following we present algorithms that, given two admissible sets represented by their gists, efficiently compute the gist of their union and of their intersection. In fact, the results of these operations are continuous but also admit a finite gist.

Union of Admissible Sets

Let A_1 and A_2 be two admissible sets and assume that their gists $\hat{A}_1$ and $\hat{A}_2$ are sorted with respect to the $\preceq_x$ relationship. Let $U = A_1 \cup A_2$, we compute its gist $\hat{U}$. Observe that all the pairs contained in $\hat{U}$ are preceded by at least one pair of either $\hat{A}_1$ or $\hat{A}_2$. Also observe that, for each pair $p \in \hat{U}$, either $p \in \hat{A}_1$ or $p \in \hat{A}_2$.

We propose a recursive algorithm, called GIST_UNION, which, given the gists $\hat{A}_1$ and $\hat{A}_2$, sorted with respect to the $\preceq_x$ relationship, builds a set S that is the gist $\hat{U}$ of $U = A_1 \cup A_2$, sorted with respect to the $\preceq_x$ relationship.

Initialize S to the empty set. Then, starting from the first pairs of the two sets, compare the two current pairs, (possibly) add to S one of the two, and update the current pairs, consuming at each comparison one pair of at least one of the two sets. When one of the two sets is empty, add to S all the pairs of the other set.

More in detail, when comparing $p_1 = \langle x_1, y_1 \rangle \in \hat{A}_1$ and $p_2 = \langle x_2, y_2 \rangle \in \hat{A}_2$, if $p_1 \nsim p_2$, then add the pair with minimum x, say p_1, to S and remove it from $\hat{A}_1$. If $p_1 \preceq p_2$, then remove p_2 from $\hat{A}_2$ without adding any pair to S. Analogously, if $p_2 \preceq p_1$ then remove p_1 from $\hat{A}_1$ without adding any pair to S.

Lemma 5.1 *Starting from the gists $\hat{A}_1$ and $\hat{A}_2$ of two admissible sets, sorted with respect to the $\preceq_x$ relationship, Algorithm* GIST_UNION *computes the gist $\hat{U}$ of $U = A_1 \cup A_2$, sorted with respect to the $\preceq_x$ relationship.*

Proof: Let S be the set of integer pairs computed by Algorithm GIST_UNION. We prove the statement by proving that (a) S is succinct; (b) $S \subset U$; and (c) $S \preceq U$. First observe that, since the pairs added to S have increasing values of x, S is sorted with respect to the $\preceq_x$ relationship.

(a) (S is succinct) We prove that the pair p_i added to S at step i is incomparable with the pairs $p_1, p_2, \ldots, p_{i-1}$ added to S at the previous steps. For $i = 1$ the statement is trivially true. Consider step $i > 1$. By Property 5.1.5 and by the fact that S is sorted with respect to the $\preceq_x$ relationship, we have that if $p_i \nsim p_{i-1}$, then $p_i \nsim p_j$, for $j = 1, \ldots, i-1$. Hence, it is sufficient to prove that $p_i = \langle x_i, y_i \rangle \nsim p_{i-1} = \langle x_{i-1}, y_{i-1} \rangle$. Suppose, without loss of generality, that $p_i \in \hat{A}_1$. If $p_{i-1} \in \hat{A}_1$, since $\hat{A}_1$ is succinct, we have $p_i \nsim p_{i-1}$. If $p_{i-1} \in \hat{A}_2$, consider the pair $p^* = \langle x^*, y^* \rangle \in \hat{A}_1$ compared with p_{i-1} when it was added to S. By construction, $p_{i-1} \nsim p^*$ and $x_{i-1} < x^*$. Since $\hat{A}_1$ is succinct and sorted with respect to the $\preceq_x$ relationship, we have that $p^* \nsim p_i$ and $x^* < x_i$. Hence, by Property 5.1.5, we have $p_i \nsim p_{i-1}$.

(b) ($S \subset U$) By construction, each pair $p \in S$ belongs to either $\hat{A}_1$ or $\hat{A}_2$. Hence, by definition, it belongs to $U = A_1 \cup A_2$.

(c) ($S \preceq U$) We show that any pair $p \in U$ is preceded by at least one pair in S. Suppose, without loss of generality, that p is preceded by at least

one pair of $\hat{A}_1$. Let $p_1 = \langle x_1, y_1 \rangle$ be the pair with the greatest value of x preceding p in $\hat{A}_1$. If $p_1 \in S$, the statement follows. If $p_1 \notin S$, since p_1 was not added to S, there exists a pair $p_2 \in \hat{A}_2$ such that $p_2 \preceq p_1 \preceq p$. Since $\hat{A}_1$ is succinct, no pair of $\hat{A}_1$ precedes p_2. Hence, $p_2 \in S$.

□

Lemma 5.2 ***(Union complexity by size)*** *Let A_1 and A_2 be two admissible sets and let $\hat{A}_1$ and $\hat{A}_2$ be their gists, sorted with respect to the $\preceq_x$ relationship. The gist $\hat{U}$ of $U = A_1 \cup A_2$, sorted with respect to the $\preceq_x$ relationship, can be computed in $O(|\hat{A}_1| + |\hat{A}_2|)$ time.*

Proof: Apply Algorithm GIST_UNION to $\hat{A}_1$ and $\hat{A}_2$. At every step the algorithm removes one pair from at least one of the two sets. Hence, at most $O(|\hat{A}_1| + |\hat{A}_2|)$ steps are performed. Since each step can be executed in $O(1)$ time, the statement follows. □

Lemma 5.3 ***(Union complexity by value)*** *Let A_1 and A_2 be two admissible sets and let $\hat{A}_1$ and $\hat{A}_2$ be their gists, sorted with respect to the $\preceq_x$ relationship. The gist $\hat{U}$ of $U = A_1 \cup A_2$, sorted with respect to the $\preceq_x$ relationship, can be computed in $O(\max(x^{\max}(\hat{A}_1), x^{\max}(\hat{A}_2)))$ time. Further, $x^{\max}(\hat{U})$ is $\min(x^{\max}(\hat{A}_1), x^{\max}(\hat{A}_2))$.*

Proof: Apply Algorithm GIST_UNION to $\hat{A}_1$ and $\hat{A}_2$. The time complexity bound follows from Lemma 5.2 and from the fact that, by Property 5.1.4, we have $|\hat{A}_1| \leq x^{\max}(\hat{A}_1)$ and $|\hat{A}_2| \leq x^{\max}(\hat{A}_2)$. For the bound on $x^{\max}(\hat{U})$, consider the last pair of $\hat{U}$, that is, by Properties 5.1.3 and 5.3, pair $\langle x^{\max}(\hat{U}), 0 \rangle$. By construction, such a pair is either $p_1 = \langle x^{\max}(\hat{A}_1), 0 \rangle \in \hat{A}_1$ or $p_2 = \langle x^{\max}(\hat{A}_2), 0 \rangle \in \hat{A}_2$. Since $\langle \min(x^{\max}(\hat{A}_1), x^{\max}(\hat{A}_2)), 0 \rangle$ precedes both p_1 and p_2, it follows that $x^{\max}(\hat{U})$ is $\min(x^{\max}(\hat{A}_1), x^{\max}(\hat{A}_2))$. □

Lemma 5.4 *Let A_j, $j = 1, \ldots, k$, be k admissible sets and let $\hat{A}_j$ be their gists, each one sorted with respect to the $\preceq_x$ relationship. The gist of $U = A_1 \cup A_2 \cup \cdots \cup A_k$, sorted with respect to the $\preceq_x$ relationship, can be computed in $O(\sum_{j=1}^{k}(x^{\max}(\hat{A}_j)))$ or, equivalently, in $O(\sum_{j=1}^{k}(|\hat{A}_j|))$ time.*

Proof: Let A_m be the admissible set such that $x^{\max}(\hat{A}_m) \leq x^{\max}(\hat{A}_j)$, for $j = 1, \ldots, k$. Starting from $U_0 = A_m$, apply GIST_UNION to U_{j-1} and A_j in order to obtain U_j, $j = 1, \ldots, k$, $j \neq m$, where $U = U_k$. By Lemma 5.3 each

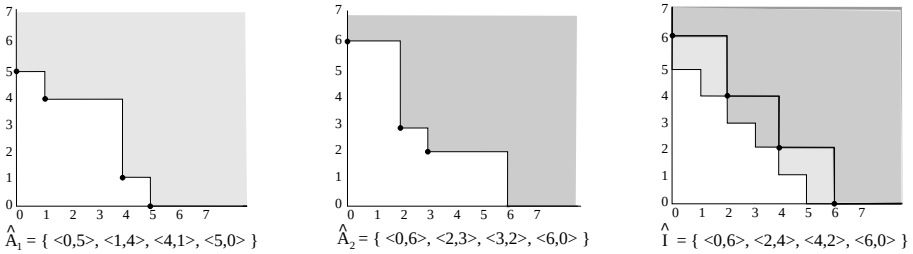


Figure 5.6: Computation of $\hat{I}$, where $I = A_1 \cap A_2$, given $\hat{A}_1$ and $\hat{A}_2$. An integer pair $\langle x, y\rangle$ belonging to a gist is represented by a black circle with coordinates (x, y).

step has time complexity $O(x^{\max}(\hat{A}_j))$ and $x^{\max}(U_j) = x^{\max}(\hat{A}_m)$. Hence, the computation can be performed in $O(\sum_{j=1}^{k}(|\hat{A}_j|))$ time. □

Intersection of Admissible Sets

Let A_1 and A_2 be two admissible sets and assume that their gists $\hat{A}_1$ and $\hat{A}_2$ are sorted with respect to the $\preceq_x$ relationship. Let $I = A_1 \cap A_2$, we compute its gist $\hat{I}$. Observe that all the pairs contained in $\hat{I}$ are preceded by at least one pair of both $\hat{A}_1$ and $\hat{A}_2$. Also, observe that $\hat{I}$ may contain some pair p such that $p \notin \hat{A}_1$ and $p \notin \hat{A}_2$. This can be seen, for example, by pairs $\langle 2, 4\rangle, \langle 4, 2\rangle \in \hat{I}$ of Fig. 5.6, and is proved in the following lemma.

Lemma 5.5 *Let A_1 and A_2 be two admissible sets and let $\hat{A}_1$ and $\hat{A}_2$ be their gists, sorted with respect to the $\preceq_x$ relationship, such that $\hat{A}_1 = \{\langle x_1, y_1\rangle, \ldots, \langle x_j, y_j\rangle, \langle x_{j+1}, y_{j+1}\rangle, \ldots, \langle x_m, y_m\rangle\}$ and $\langle x, y\rangle \in \hat{A}_2$. If $x_j \leq x < x_{j+1}$, then $\langle x, \max(y, y_j)\rangle \in I = A_1 \cap A_2$.*

Proof: For each pair $\langle x_k, y_k\rangle \in \hat{A}_1$, with $k = 1, \ldots, j$, by $x_k \leq x$, there exist infinite pairs $\langle x, y_k + m\rangle \in A_1$, with $m \geq 0$. By Property 5.1.3, $\langle x, y_j\rangle \preceq \langle x, y_k + m\rangle$, for any $k = 1, \ldots, j$ and $m \geq 0$. Hence, by $x_j < x$, we have $\langle x_j, y_j\rangle \preceq \langle x, max(y, y_j)\rangle$. Since $\langle x, \max(y, y_j)\rangle$ is preceded by $\langle x_j, y_j\rangle \in \hat{A}_1$ and by $\langle x, y\rangle \in \hat{A}_2$, the statement follows. □

We propose a recursive algorithm, called GIST_INTERSECTION, which, given the gists $\hat{A}_1$ and $\hat{A}_2$, sorted with respect to the $\preceq_x$ relationship, builds a set S that is the gist $\hat{I}$ of $I = A_1 \cap A_2$, sorted with respect to the $\preceq_x$ relationship.

Algorithm GIST_INTERSECTION is analogous to GIST_UNION. The only difference is in the way the pairs are selected to be added to S and to be removed from $\hat{A}_1$ and $\hat{A}_2$.

More in detail, consider the two current pairs $p_1 = \langle x_1, y_1 \rangle \in \hat{A}_1$ and $p_2 = \langle x_2, y_2 \rangle \in \hat{A}_2$. Suppose $x_1 = x_2$. Let $p^* = \langle x_1, \max(y_1, y_2) \rangle$. Remove p_1 and p_2 from $\hat{A}_1$ and $\hat{A}_2$, respectively, and if p^* is not preceded by the last pair added to S, then add p^* to S. Suppose $x_1 < x_2$, the case $x_1 > x_2$ being analogous. Consider the pair $p^* = \langle x_1, \max(y_1, y_2') \rangle$, where $\langle x_2', y_2' \rangle$ is the last pair removed from $\hat{A}_2$. Remove p_1 from $\hat{A}_1$ and if p^* is not preceded by the last pair added to S, then add p^* to S.

Lemma 5.6 *Starting from the gists $\hat{A}_1$ and $\hat{A}_2$ of two admissible sets A_1 and A_2, sorted with respect to the $\preceq_x$ relationship, Algorithm* GIST_INTERSECTION *computes the gist $\hat{I}$ of $I = A_1 \cap A_2$, sorted with respect to the $\preceq_x$ relationship.*

Proof: Let S be the set of integer pairs computed by Algorithm GIST_INTERSECTION. First observe that the pairs added to S have increasing values of x. We prove the statement by proving that (a) S is succinct; (b) $S \subset I$; and (c) $S \preceq I$.

(a) (S is succinct) Analogously to the proof of point (a) of Lemma 5.1, it is sufficient to prove that $p_i \nsim p_{i-1}$, where p_i is the pair added to S at step i. Since p_i is added to S only if it is not preceded by p_{i-1} and, by Property 5.1.3, it holds $p_i \not\preceq p_{i-1}$, the statement follows.

(b) ($S \subset I$) Consider the pairs $p_1 = \langle x_1, y_1 \rangle \in \hat{A}_1$ and $p_2 = \langle x_2, y_2 \rangle \in \hat{A}_2$ compared at step i when adding pair p_i to S. If $x_1 = x_2$, then $p_i = \langle x_1, \max(y_1, y_2) \rangle$ belongs to both A_1 and A_2, since it is preceded by both p_1 and p_2. If $x_1 \neq x_2$, then $p_i = \langle x_1, \max(y_1, y_2') \rangle$, where $\langle x_2', y_2' \rangle$ is the last pair removed from $\hat{A}_2$. By Lemma 5.5, $p_i \in A_1 \cap A_2$.

(c) ($S \preceq I$) We show that any pair $p \in A_1 \cap A_2$ is preceded by at least one pair in S. Let $p_1 = \langle x_1, y_1 \rangle$ and $p_2 = \langle x_2, y_2 \rangle$ be the two pairs with the greatest value of x preceding p in $\hat{A}_1$ and $\hat{A}_2$, respectively. If $x_1 = x_2$, let $p_i = \langle x_1, \max(y_1, y_2) \rangle$. If p_i is not preceded by the last pair added to S, p_i is added to S. If $x_1 < x_2$ (case $x_1 > x_2$ being analogous), Algorithm GIST_INTERSECTION adds a suitable pair to S, removes p_1 from $\hat{A}_1$, and compares $p_1^* = \langle x_1^*, y_1^* \rangle$ and $p_2 = \langle x_2, y_2 \rangle$, where p_1^* is the

pair immediately following p_1 in $\hat{A}_1$. Observe that, since p_1 and p_2 were chosen to be the two pairs with greatest x preceding p in $\hat{A}_1$ and $\hat{A}_2$, we have $x_1^* > x_2$. If $p_{i+1} = \langle x_2, \max(y_2, y_1)\rangle$ is not preceded by the last pair added to S, then p_{i+1} is added to S. Hence, in all cases, a pair preceding p is added to S. In fact, since $x \geq x_1$, $y \geq y_1$, and $y \geq y_2$, we have $p_i \preceq p$, and since $x \geq x_2$, $y \geq y_1$, and $y \geq y_2$, we have $p_{i+1} \preceq p$.

□

The following lemma can be proved analogously to Lemma 5.2.

Lemma 5.7 ***(Intersection complexity by size)*** *Let A_1 and A_2 be two admissible sets and let $\hat{A}_1$ and $\hat{A}_2$ be their gists, sorted with respect to the $\preceq_x$ relationship. The gist $\hat{I}$ of $I = A_1 \cap A_2$, sorted with respect to the $\preceq_x$ relationship, can be computed in $O(|\hat{A}_1| + |\hat{A}_2|)$ time.*

Lemma 5.8 ***(Intersection complexity by value)*** *Let A_1 and A_2 be two admissible sets and let $\hat{A}_1$ and $\hat{A}_2$ be their gists, sorted with respect to the $\preceq_x$ relationship. The gist $\hat{I}$ of $I = A_1 \cap A_2$, sorted with respect to the $\preceq_x$ relationship, can be computed in $O(\min(x^{\max}(\hat{A}_1), x^{\max}(\hat{A}_2))$ time. Further, $x^{\max}(\hat{I})$ is $\max(x^{\max}(\hat{A}_1), x^{\max}(\hat{A}_2))$.*

Proof: Apply Algorithm GIST_INTERSECTION to $\hat{A}_1$ and $\hat{A}_2$. At each step the algorithm removes the pair with smallest x from $\hat{A}_1$ or $\hat{A}_2$. Hence, a pair $p = \langle x, y\rangle$ is removed from $\hat{A}_1$ or $\hat{A}_2$ after at most x steps. Let $\hat{A}_m$ ($\hat{A}_M$) be the set with minimum (maximum) value of $x^{\max}$ between $\hat{A}_1$ and $\hat{A}_2$. After $O(\min(x^{\max}(\hat{A}_1), x^{\max}(\hat{A}_2))$ steps the set $\hat{A}_m$ is empty. Using an appropriate data structure, all the remaining pairs of the non-empty set can be added to $\hat{I}$ in $O(1)$ time, yielding the time complexity bound. The second part of the statement follows from the fact that the last pair of $\hat{I}$ is the last pair of $\hat{A}_M$. □

Analogously to Lemma 5.4, the following lemma shows that the gist of the intersection of k sets of integer pairs can be computed in time linear in the sum of the sizes of their gists.

Lemma 5.9 *Let A_j, for $j = 1, \ldots, k$, be k admissible sets and let $\hat{A}_j$ be their gists, each one sorted with respect to the $\preceq_x$ relationship. The gist $\hat{I}$ of $I = A_1 \cap A_2 \cap \cdots \cap A_k$, sorted with respect to the $\preceq_x$ relationship, can be computed in $O(\sum_{j=1}^{k}(x^{\max}(\hat{A}_j)))$ or, equivalently, in $O(\sum_{j=1}^{k}(|\hat{A}_j|))$ time.*

Proof: Let A_M be the admissible set such that $x^{\max}(\hat{A}_M) \geq x^{\max}(\hat{A}_j)$, for $j = 1, \ldots, k$. Starting from $I_0 = A_M$, apply GIST_INTERSECTION to I_{j-1} and

A_j in order to obtain I_j, $j = 1, \ldots, k$, $j \neq M$, where $I = I_k$. By Lemma 5.8 each step has time complexity $O(x^{\max}(\hat{A}_j))$ and $x^{\max}(I_j) = x^{\max}(\hat{A}_M)$. Hence, the computation can be performed in $O(\sum_{j=1}^{k}(|\hat{A}_j|))$ time. □

5.5 Computing a Minimum-Depth Embedding of a Biconnected Planar Graph

In order to determine the minimum k for which a biconnected planar graph G admits an embedding with depth k, we apply for each edge e of G the algorithm presented in this section, which determines the minimum k for which G admits an embedding of depth k with e on the external face.

Such a computation is performed by means of a bottom-up traversal of the SPQR-tree of G whose purpose is to label each virtual edge e_i, corresponding to node μ_i, with suitable values that describe the properties related to the depth of all possible embeddings of the pertinent graph G_{μ_i}.

Such values, called *depth descriptors*, are:

- The gist $\hat{A}(\mu_i)$ of the admissible set of μ_i
- The distance between $f_l^{\mu_i}$ and $f_r^{\mu_i}$ in the (u_i, v_i)-dual of G_{μ_i}, which is called the *thickness* of μ_i and is denoted by $t(\mu_i)$.

In [BM90], where the concept of thickness was also used, it is shown that $t(\mu_i)$ is independent of the embedding of G_{μ_i}. Hence, $t(\mu_i)$ can also be defined as the distance between $f_l^{\mu_i}$ and $f_r^{\mu_i}$ in the (u_i, v_i)-dual of $sk(\mu_i)$, where the edges have been suitably weighted. Namely, each edge of the (u_i, v_i)-dual of $sk(\mu_i)$ that corresponds to a virtual edge e_ν of $sk(\mu_i)$ representing a child component ν is given a weight that is equal to the thickness of ν.

At the end of the bottom-up traversal of the SPQR-tree $\mathcal{T}$, the unique child component of the root e of $\mathcal{T}$ is labeled with the gist of the admissible set of G, describing all possible depths of the embeddings of G with e on the external face. From such an admissible set an optimal pair can be selected to determine the minimum depth k when e is on the external face. The minimum depth of G can be obtained by performing such a computation for each edge e of G.

In order to actually compute an optimal embedding of G, the bottom-up traversal has to be refined by labeling each virtual edge e_i, corresponding to node μ_i, with additional embedding descriptors (see Section 5.5) meant to describe how the components must be combined together in order to obtain an

embedding satisfying each pair of the gist of the admissible set. Such descriptors are used in a subsequent top-down traversal of $\mathcal{T}$, rooted at the edge e that yielded the minimum value of k, to select a suitable embedding for the skeleton of each node of $\mathcal{T}$, producing an embedding of G with minimum depth k.

Labeling an SPQR-tree with Depth Descriptors

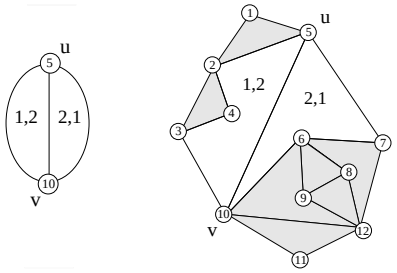
During the bottom-up traversal of $\mathcal{T}$, for each component μ, we compute its thickness $t(\mu)$ and its gist $\hat{A}(\mu)$ based on the analogous values of its children.

The computation of $t(\mu)$ is easy and, sometimes, trivial. The computation of $\hat{A}(\mu)$, instead, is much more complex. The most difficult case arises when μ is a parallel node. Here, we sketch the general strategy to compute $\hat{A}(\mu)$, especially when μ is a series or rigid component. Details on the computation of depth descriptors for series, rigid, and parallel cases can be found in the following three subsections. For the series and rigid cases, our strategy is based on the fact that the set of all embeddings of the pertinent graph G_μ of μ can be suitably partitioned, and each block of the partition can be separately analyzed. For the parallel case, instead, our strategy is based on the exploration of a bounded-size subset of the set of all possible embeddings of G_μ which maintains the same admissible set of all embeddings of G_μ.

Let μ be a node of $\mathcal{T}$ and let $sk(\mu)$ be its skeleton. If μ is a series, then $sk(\mu)$ has a unique embedding Γ^1_μ; if μ is a rigid, then $sk(\mu)$ admits two embeddings Γ^1_μ and Γ^2_μ; and if μ is a parallel with k child components, then $sk(\mu)$ admits $k!$ embeddings Γ^h_μ, with $h = 1, \ldots, k!$. Each embedding of G_μ is compatible with exactly one embedding Γ^j_μ of $sk(\mu)$. Hence, the embeddings of $sk(\mu)$ induce a partition on the embeddings of G_μ. For series and rigid nodes, in order to compute $A(\mu)$ through all possible embeddings of G_μ, we first compute the admissible sets $A^j(\mu)$, restricted to those embeddings of G_μ corresponding to a single embedding Γ^j_μ of $sk(\mu)$, and then perform their union.

Given an embedding Γ_{G_μ} of the pertinent graph G_μ of μ we distinguish two types of faces. We call *children faces* the faces of Γ_{G_μ} that are also faces of some Γ_{G_ν}, with ν child of μ, and *skeleton faces* all the other faces. Essentially, "shrinking" each pertinent graph of the children of μ into a single (virtual) edge, the skeleton faces of Γ_{G_μ} become the faces of an embedding Γ^j_μ of $sk(\mu)$.

Observe that, once the embedding Γ^j_μ of $sk(\mu)$ has been fixed, the distances of each skeleton face of any embedding of Γ_{G_μ} from f^μ_l and f^μ_r depend on the values $t(\nu_1), \ldots, t(\nu_k)$ only, which, in turn, are independent on the embedding of the child components of μ. Hence, each face f of Γ^j_μ can be labeled with such



(a) (b)

Figure 5.7: (a) Embedding $\Gamma^1_{P_1}$ of the skeleton of parallel component P_1. (b) An embedding Γ_{P_1} of P_1 compatible with $\Gamma^1_{P_1}$. Skeleton faces are drawn white and children faces are drawn grey. Skeleton faces are labeled with their distances from the external faces.

distances, that are called *left* and *right depths* and denoted by $d_l(f)$ and $d_r(f)$. Fig. 5.7(a) shows an embedding $\Gamma^1_{P_1}$ of the skeleton of parallel component P_1 and Fig. 5.7(b) shows an embedding Γ_{P_1} of the pertinent graph of P_1 compatible with $\Gamma^1_{P_1}$. Notice that skeleton faces, which are drawn white, have the same values of depth in $\Gamma^1_{P_1}$ and in Γ_{P_1}.

Definitions analogous to those given for G_μ can be given for $sk(\mu)$. In particular, we say that Γ^j_μ *satisfies* the pair of non-negative integers $\langle x, y\rangle$ if it is possible to find a partition of its internal faces into two sets, denoted by F_l and F_r, such that each face $f \in F_l$ has $d_l(f) \leq x$ and each face $f \in F_r$ has $d_r(f) \leq y$. The infinite set of integer pairs satisfied by Γ^j_μ is the *admissible set of* Γ^j_μ, and is denoted by $A(\Gamma^j_\mu)$.

Once the embedding Γ^j_μ of $sk(\mu)$ has been fixed, the admissible set $A^j(\mu)$, i.e., the admissible set of μ restricted to the embeddings compatible with Γ^j_μ, can be computed starting from $A(\Gamma^j_\mu)$ and from the depths of the skeleton faces, together with the admissible set $A(\nu_i)$ of each child component ν_i, with $i = 1, \ldots, \delta(\mu)$.

Namely, μ satisfies the integer pair $\langle x, y\rangle$ if $\langle x, y\rangle \in A(\Gamma^j_\mu)$ and each child component ν_i satisfies a pair $\langle x^i, y^i\rangle$ such that:

- $x^i + d_l(f_l^{\nu_i}) \leq x$ or $x^i + d_r(f_l^{\nu_i}) \leq y$, and
- $y^i + d_r(f_r^{\nu_i}) \leq y$ or $y^i + d_l(f_r^{\nu_i}) \leq x$.

Hence, in order to obtain $A^j(\mu)$, we compute for each child component ν_i of μ the set of integer pairs that are satisfied by ν_i when inserted into Γ^j_μ, that is, the set of integer pairs that verify the conditions above.

Namely, let μ be a node of the SPQR-tree $\mathcal{T}$, let Γ^j_μ be an embedding of $sk(\mu)$, let ν be a child of μ, and let $\langle x, y\rangle$ be a pair of non-negative integers. Node ν *satisfies* $\langle x, y\rangle$*, nested into* Γ^j_μ, if the pertinent graph G_μ of μ admits an embedding Γ_{G_μ}, compatible with Γ^j_μ, where it is possible to find a partition of the set of the children faces corresponding to the internal faces of ν into two sets, denoted by F_l and F_r, such that all faces in F_l have distance from f^μ_l less or equal than x and all faces in F_r have distance from f^μ_r less or equal than y. In Fig. 5.8 it is shown how component S_2 satisfies, nested into $\Gamma^j(P_1)$, pairs $\langle 0, 2\rangle$, $\langle 3, 2\rangle$, and $\langle 4, 0\rangle$, with the corresponding partitions of its internal faces. Each internal face of S_2 is labeled with a pair of integers representing its distance from the left and the right external faces of $\Gamma^j(P_1)$, respectively.

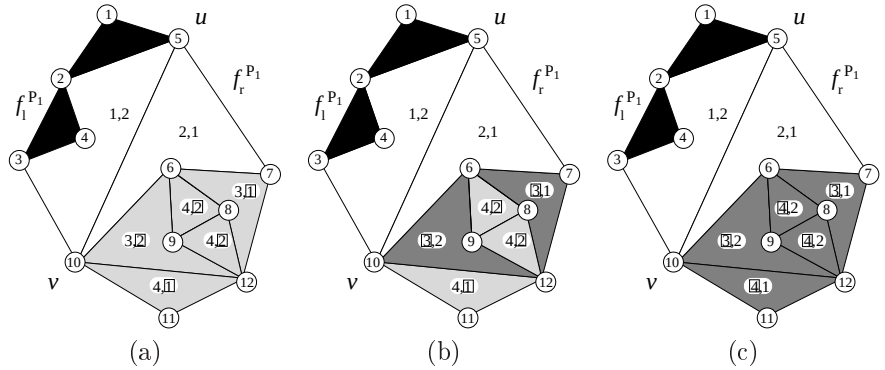


Figure 5.8: An embedding of component S_2, nested into $\Gamma^j(P_1)$. Figures (a), (b), and (c) represent three possible partitions of the internal faces of S_2. Dark-shaded faces belong to F_l, light-shaded faces belong to F_r, skeleton faces are drawn white, and children faces not belonging to S_2 are drawn black. Each face is labeled with two integers representing its distance from $f_l^{P_1}$ and $f_r^{P_1}$, respectively. Labels lying inside a square are those used in the corresponding partition. The pictures show that S_2 satisfies, nested into $\Gamma^j(P_1)$, pairs: $\langle 0, 2\rangle$ (a), $\langle 3, 2\rangle$ (b), and $\langle 4, 0\rangle$ (c).

The infinite set of integer pairs satisfied by ν, nested into Γ^j_μ, is the *ad-*

missible set of ν *into* Γ_μ^j, and is denoted by $A(\nu|\Gamma_\mu^j)$. The gist of $A(\nu|\Gamma_\mu^j)$ is denoted by $\hat{A}(\nu|\Gamma_\mu^j)$ and assumed ordered with respect to the $\preceq_x$ relationship.

Lemma 5.10 *Given an embedding* Γ_μ^j *of* $sk(\mu)$*, the admissible set* $A^j(\mu)$ *of* G_μ *(restricted to those embeddings of* G_μ *corresponding to* Γ_μ^j*) can be obtained by intersecting the* $\delta(\mu)$ *sets* $A(\nu_i|\Gamma_\mu^j)$*, for* $i = 1, \ldots, \delta(\mu)$*, and* $A(\Gamma_\mu^j)$*.*

Proof: The proof is based on the fact that the distances between the internal faces of the embedding of a component ν_k and the external faces f_l^μ and f_r^μ of μ are independent on the embedding of other child components of μ.

We first show that, given Γ_μ^j of $sk(\mu)$, a pair belonging to $A^j(\mu)$ of G_μ also belongs to $A(\nu_i|\Gamma_\mu^j)$ and to $A(\Gamma_\mu^j)$. Second, we show that if a pair belongs to $A(\nu_i|\Gamma_\mu^j)$ and to $A(\Gamma_\mu^j)$, then it also belongs to $A^j(\mu)$ of G_μ.

Let $p = \langle x, y\rangle$ be a pair of non-negative integers belonging to $A^j(\mu)$ of G_μ and let $A(\nu_k|\Gamma_\mu^j)$ be the admissible set of component ν_k nested into Γ_μ^j. Since $p \in A^j(\mu)$, there exists an embedding Γ_{G_μ} of G_μ, coherent with the embedding Γ_μ^j of $sk(\mu)$, whose internal faces can be partitioned into two sets F_l and F_r such that faces in F_l are at distance less or equal than x from f_l^μ and faces in F_r are at distance less or equal than y from f_r^μ. In order to show that p belongs to $A(\nu_k|\Gamma_\mu^j)$, it suffices to observe that the faces of the embedding Γ_{G_μ} that also belong to ν_k can be partitioned into two sets $F_l' \subseteq F_l$ and $F_r' \subseteq F_r$ such that faces in F_l' are at distance less or equal than x from f_l^μ and faces in F_r' are at distance less or equal than y from f_r^μ. Analogously, it can be shown that p belongs to $A(\Gamma_\mu^j)$, since from F_l and F_r a suitable partition of the skeleton faces can be found as required by the definition of $A(\Gamma_\mu^j)$.

Conversely, let $p = \langle x, y\rangle$ be a pair of non-negative integers belonging to $A(\nu_i|\Gamma_\mu^j)$ and to $A(\Gamma_\mu^j)$. An embedding Γ_{G_μ} of G_μ can be obtained from the embeddings of ν_i and $sk(\mu)$ that satisfy p. □

As said above, the admissible set $A(\mu)$ can be easily obtained as the union of the admissible sets $A^j(\mu)$ computed for any embedding Γ_μ^j of $sk(\mu)$.

The Series Case.

Let μ be an S-node with children ν_i, for $i = 1, \ldots, \delta(\mu)$, and let $n(\nu_i)$ be the number of vertices of ν_i. The computation of the thickness of μ is addressed in the following lemma.

Lemma 5.11 *The thickness* $t(\mu) = \min_i\{t(\nu_i)\}$*, for* $i = 1, \ldots, \delta(\mu)$*, can be computed in* $O(\delta(\mu))$ *time.*

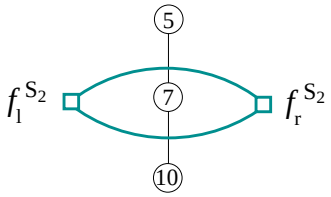


Figure 5.9: The unique embedding of $sk(S_2)$ and its (u, v)-dual.

Proof: As shown in Fig. 5.9 for a series of two child components, the (u, v)-dual graph of $sk(\mu)$ is made up of two vertices, corresponding to f_l^μ and f_r^μ, and $O(\delta(\mu))$ edges connecting them, each one corresponding to a child component ν_i and associated with a weight that is $t(\nu_i)$. Since, by definition, $t(\mu)$ is the distance between f_l^μ and f_r^μ in such a (u, v)-dual graph, the statement follows. □

As for the admissible set of μ, in the series case $sk(\mu)$ has exactly one embedding and such an embedding has no internal face. Hence, in order to compute $\hat{A}(\mu)$, it is not necessary to compute $A(\Gamma_\mu^1)$ and it is sufficient, by Lemma 5.10, to intersect the gists $\hat{A}(\nu_i|\Gamma_\mu^1)$ of the admissible sets of the child components ν_i nested into Γ_μ^1.

We propose an algorithm, called NESTED_SERIES, which, given an S-node μ and one of its children ν, suitably builds a set S starting from $\hat{A}(\nu)$ and $t(\mu)$, and we show that $S = \hat{A}(\nu|\Gamma_\mu^1)$. The algorithm starts initializing S with $\hat{A}(\nu)$. Observe that, by Property 5.3, $\hat{A}(\nu)$ contains the two pairs $p_{first} = \langle 0, y_{max}\rangle$ and $p_{last} = \langle x_{max}, 0\rangle$, with $y_{max} = x_{max}$. For each pair $p^k = \langle x_k, y_k\rangle$ of $\hat{A}(\nu)$, define $p^k_{first} = \langle 0, max(y_k, x_k + t(\mu))\rangle$. Denote by $\overline{p}_{first}$ the p^k_{first} with minimum y and by $\overline{p}_{last}$ the pair obtained from $\overline{p}_{first}$ swapping the two elements x and y. If $\overline{p}_{first} \preceq p_{first}$, then insert $\overline{p}_{first}$ into S and remove from S any pair p^* such that $\overline{p}_{first} \preceq p^*$. If $\overline{p}_{last} \preceq p_{last}$, then append $\overline{p}_{last}$ to S and remove from S any pair p^* such that $\overline{p}_{last} \preceq p^*$.

Property 5.6 *The set S computed by Algorithm* NESTED_SERIES *is succinct.*

Proof: Set S is initialized to $\hat{A}(\nu)$, that is succinct, and, when the two pairs $\overline{p}_{first}$ and $\overline{p}_{last}$ are added, no pair of S precedes them and all pairs of S preceded by them are removed from S. □

Algorithm 3 NESTED_SERIES

Require: An S-node μ, with its thickness $t(\mu)$, and one of its children ν, with its gist $\hat{A}(\nu)$.

Ensure: The gist $\hat{A}(\nu|\Gamma^1_\mu)$ of the admissible set of ν nested into Γ^j_μ.

1: $S = \hat{A}(\nu)$;
2: $p_{first} = \hat{A}(\nu)$.getFirst();
3: **for all** $p^k = \langle x_k, y_k \rangle \in \hat{A}(\nu)$ **do**
4: $\quad p^k_{first} = \langle 0, \max(y_k, x_k + t(\mu)) \rangle$
5: $\quad$ **if** $p^k_{first} \preceq \overline{p}_{first}$ **then**
6: $\quad\quad \overline{p}_{first} = p^k_{first}$;
7: $\quad$ **end if**
8: **end for**
9: **if** $\overline{p}_{first} \preceq p_{first}$ **then**
10: $\quad S$.addFirst($\overline{p}_{first}$);
11: $\quad \overline{p}_{last} = \overline{p}_{first}$.swapElements();
12: $\quad S$.addLast($\overline{p}_{last}$);
13: $\quad$ **for all** $p^* \neq \overline{p}_{first} \in S$ and $p^* \neq \overline{p}_{last} \in S$ **do**
14: $\quad\quad$ **if** $\overline{p}_{first} \preceq p^*$ or $\overline{p}_{last} \preceq p^*$ **then**
15: $\quad\quad\quad S$.remove(p^*);
16: $\quad\quad$ **end if**
17: $\quad$ **end for**
18: **end if**
19: **return** S;

Property 5.7 *The set S computed by Algorithm* NESTED_SERIES *is a subset of $A(\nu|\Gamma^1_\mu)$.*

Proof: Consider a pair $p = \langle x, y \rangle \in S$. Two are the cases: Either p is also in $\hat{A}(\nu)$ or not. If $p \in \hat{A}(\nu)$, then $p \in A(\nu|\Gamma^1_\mu)$, since μ is an S-node. If $p \notin \hat{A}(\nu)$, then p is either $\overline{p}_{first}$ or $\overline{p}_{last}$. First, consider the case $p = \overline{p}_{first}$. We show that $p \in A(\nu|\Gamma^1_\mu)$ as follows. Consider the pair $p^k = \langle x_k, y_k \rangle$ such that $p^k_{first} = \overline{p}_{first}$. Since $p^k \in \hat{A}(\nu)$, there exists an embedding Γ^k_ν such that the set of the internal faces of Γ^k_ν can be partitioned into two sets F^k_l and F^k_r such that all faces in F^k_l have distance from f^ν_l less or equal than x_k and all faces in F^k_r have distance from f^ν_r less or equal than y_k.

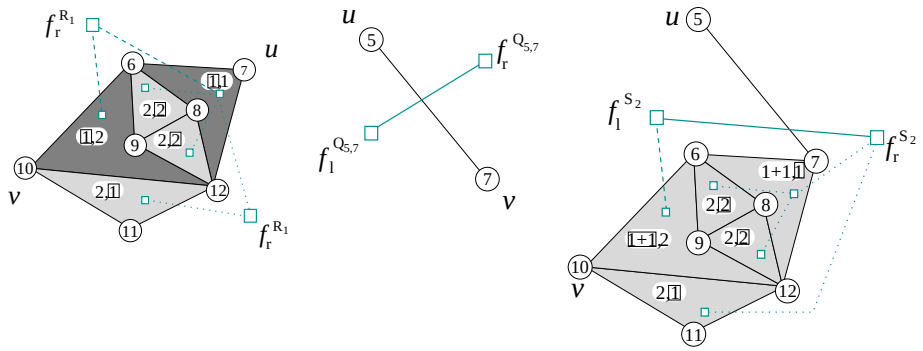


Figure 5.10: (a) Component R_1 satisfies pair $p^k = \langle x_k, y_k \rangle = \langle 1, 2 \rangle$. (b) Edge $(5, 7)$ has thickness 1. (c) Component R_1 satisfies, nested into $\Gamma^J_{S_2}$, pair $p^k_{first} = \langle 0, max(y_k, x_k + t(\mu)) \rangle = \langle 0, max(2, 1 + 1) \rangle = \langle 0, 2 \rangle$. Dark-shaded faces are in F^k_l and light-shaded faces are in F^k_r. Each face is labeled with two integers representing its distance from left and right external face, respectively. Labels lying inside a square are those used in the corresponding partition. Dashed edges are those used to reach the left external face, while dotted edges are those used to reach the right external face.

Consider any embedding $\Gamma^*_{G_\mu}$ such that $\Gamma^*_{G_\mu}$ restricted to ν is Γ^k_ν. Each face of $\Gamma^*_{G_\mu}$ internal to ν is at distance less or equal than $max(y_k, x_k + t(\mu))$ from f^μ_r in G_μ. In fact, faces in F^k_l are at distance less or equal than x_k from f^ν_l which, in turn, is at distance $t(\mu)$ from f^μ_r, and faces in F^k_r are at distance less or equal than y_k from f^μ_r. It follows that $p^k_{first} = \langle 0, max(y_k, x_k + t(\mu)) \rangle$ belongs to $A(\nu|\Gamma^1_\mu)$. See Fig. 5.10 for an example. The proof for the case $p = \overline{p}_{last}$ is analogous. □

Property 5.8 *The set S computed by Algorithm* NESTED_SERIES *precedes $A(\nu|\Gamma^1_\mu)$.*

Proof: Since $\hat{A}(\nu|\Gamma^1_\mu) \preceq A(\nu|\Gamma^1_\mu)$, it is sufficient to show that $S \preceq \hat{A}(\nu|\Gamma^1_\mu)$. Suppose, for a contradiction, that there exists a pair $p = \langle x, y \rangle$ such that $p \in \hat{A}(\nu|\Gamma^1_\mu)$ and there is no pair $p' \in S$ such that $p' \preceq p$. Since $p \in \hat{A}(\nu|\Gamma^1_\mu)$, there exists an embedding Γ_{G_μ} such that all faces of ν can be partitioned into

two sets F_l and F_r such that all faces in F_l are at distance less or equal than x from f_l^μ and all faces in F_r are at distance less or equal than y from f_r^μ.

First, suppose that $x, y \neq 0$. By Property 5.4, there exists at least one face f in F_l which is at distance x from f_l^μ and at distance greater than y from f_r^μ. Consider a path $\gamma(f, f_l^\mu)$ of minimum length x from f to f_l^μ. We show that $f_r^\mu \notin \gamma(f, f_l^\mu)$. Suppose, for a contradiction, that $f_r^\mu \in \gamma(f, f_l^\mu)$. We split $\gamma(f, f_l^\mu)$ into a subpath $\gamma(f, f_r^\mu)$ of length greater than y and a subpath $\gamma(f_r^\mu, f_l^\mu)$ of length $t(\mu)$. Since $|\gamma(f, f_l^\mu)| = x$ and $|\gamma(f, f_r^\mu)| > y$, it follows that $t(\mu) < x - y$. Hence, each face in F_r, which is at distance at most y from f_r^μ, is also at distance at most x from f_l^μ. This implies that the two sets $F_l \cup F_r$ and $\emptyset$ can be used to show that Γ_{G_μ} satisfies pair $\langle x, 0\rangle$. This is a contradiction since $\hat{A}(\nu|\Gamma_\mu^1)$ is succinct and contains $p = \langle x, y\rangle$. Therefore, no path of minimum length from a face $f \in F_l$ to f_l^μ contains f_r^μ. Analogously, no path of minimum length from a face $f \in F_r$ to f_r^μ contains f_l^μ. It follows that $p \in \hat{A}(\nu)$, in fact Algorithm NESTED_SERIES added p to S when S was initialized to $\hat{A}(\nu)$. Since, by construction, $S \preceq \hat{A}(\nu)$, there exists a pair $p' \in S$ such that $p' \preceq p$, a contradiction.

Now, suppose $p = \langle x, 0\rangle$. Two are the cases: Either $p \in \hat{A}(\nu)$ or not. In the first case, by construction, there exists a pair $p' \in S$ such that $p' \preceq p$, contradicting the hypothesis that there is no pair in S preceding p. If $p \notin \hat{A}(\nu)$, then for at least a face $f \in F_l$ the minimum length path $\gamma(f, f_l^\mu)$ to f_l^μ passes through f_r^μ and traverses a child of μ different from ν. It follows that the pair $p' = \langle x, y\rangle$ with $y = x - t(\mu)$ belongs to $A(\nu)$. Hence, by construction, there exists a pair $\overline{p} \in S$ that precedes $\overline{p}_{last} \preceq \langle max(x, y + t(\mu)), 0\rangle = \langle max(x, x), 0\rangle = \langle x, 0\rangle = p$, contradicting the hypothesis that there is no pair in S preceding p.

Analogous considerations show that if $p = \langle 0, y\rangle$, then there exists a pair $\overline{p} \in S$ that precedes $\overline{p}_{first} \preceq p$, contradicting the hypothesis. □

Lemma 5.12 *Starting from $\hat{A}(\nu_i)$ and $t(\mu)$, Algorithm* NESTED_SERIES *computes $\hat{A}(\nu_i|\Gamma_\mu^1)$ in time $O(n(\nu_i))$.*

Proof: The set S computed by Algorithm NESTED_SERIES is such that $S \preceq A(\nu_i|\Gamma_\mu^1)$ (Property 5.8) and $S \subseteq A(\nu_i|\Gamma_\mu^1)$ (Property 5.7). It follows that S reduces $A(\nu_i|\Gamma_\mu^1)$. Further, by Property 5.6, S is succinct. Hence, S is $\hat{A}(\nu_i|\Gamma_\mu^1)$. The time complexity bound follows from the fact that the construction of the pairs $\overline{p}_{first}$ and $\overline{p}_{last}$, as well as their insertion into S, is performed in linear time with respect to the cardinality of $\hat{A}(\nu_i)$ because in both cases each element of the set is considered only once. □

Lemma 5.13 *Starting from $\hat{A}(\nu_i)$ and $t(\nu_i)$, for $i = 1, \ldots, \delta(\mu)$, the gist $\hat{A}(\mu)$ can be computed in time $O(\sum_{i=1}^{\delta(\mu)} n(\nu_i))$.*

Proof: By Lemma 5.11, the thickness $t(\mu)$ can be computed in $O(\delta(\mu))$ time. By Lemma 5.12, given $\hat{A}(\nu_i)$ and $t(\mu)$, the gists $\hat{A}(\nu_i|\Gamma_\mu^1)$ of the admissible sets of ν_i nested into Γ_μ^1 can be computed in $O(\sum_{i=1}^{\delta(\mu)} n(\nu_i))$ total time. By Lemma 5.9, the intersection of such sets can be computed in $O(\sum_{i=1}^{\delta(\mu)} n(\nu_i))$ time. □

The Rigid Case.

Let μ be an R-node with children ν_i, for $i = 1, \ldots, \delta(\mu)$, and let $n(\nu_i)$ be the number of vertices of ν_i. The computation of the thickness is addressed by the following lemma.

Lemma 5.14 *The thickness $t(\mu)$ can be computed in $O(\delta(\mu))$ time.*

Proof: The (u, v)-dual graph of $sk(\mu)$ is a triconnected component with the two vertices corresponding to f_l^μ and f_r^μ on the external face (see, for example, Fig. 5.3). By definition, $t(\mu)$ is the distance between f_l^μ and f_r^μ in such a (u, v)-dual graph and can be computed in $O(\delta(\mu))$ performing a shortest path algorithm between f_l^μ and f_r^μ [Tho99]. □

The remaining part of this section is devoted to the computation of the admissible set of μ. In the rigid case, since $sk(\mu)$ is a 3-connected component, it admits exactly two embeddings, Γ_μ^1 and Γ_μ^2, which only differ for a flipping around their poles. For example, Figs. 5.11(a) and 5.11(b) show the two embeddings $\Gamma_{R_1}^1$ and $\Gamma_{R_1}^2$ of the skeleton of a rigid component R_1. Since the two embeddings of $sk(\mu)$ are symmetrical, it is possible to consider one of the two embeddings only, say Γ_μ^1, compute the admissible set $A^1(\mu)$ of μ restricted to Γ_μ^1, and obtain the admissible set $A^2(\mu)$ of μ restricted to Γ_μ^2 by swapping, for each pair of $A^1(\mu)$, elements x and y. The gist $\hat{A}(\mu)$ is given by the union of the two sets. By Lemma 5.10, $A^1(\mu)$ can be obtained by intersecting the gist $\hat{A}(\Gamma_\mu^1)$ of the admissible set of $sk(\mu)$ and the gists $\hat{A}(\nu_i|\Gamma_\mu^1)$ of the admissible sets of the $\delta(\mu)$ child components ν_i nested into Γ_μ^1. Fig. 5.11(c) shows an embedding of the pertinent graph G_{R_1} compatible with $\Gamma_{R_1}^1$.

In order to compute $\hat{A}(\nu_i|\Gamma_\mu^1)$ and $\hat{A}(\Gamma_\mu^1)$ it is useful to label each face f of Γ_μ^1 with its depths $d_l(f)$ and $d_r(f)$, as shown in Fig. 5.11. This can be done in

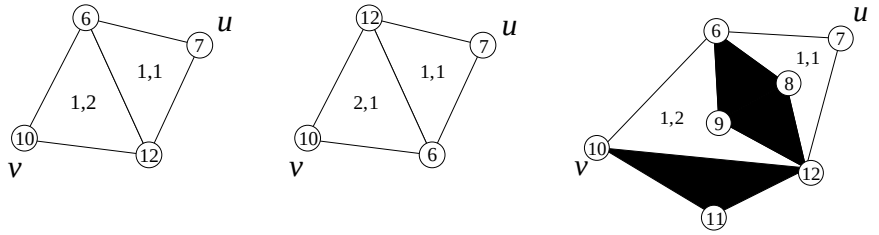


Figure 5.11: (a) Embedding $\Gamma^1_{R_1}$ of $sk(R_1)$. (b) Embedding $\Gamma^2_{R_1}$ of $sk(R_1)$, differing from $\Gamma^1_{R_1}$ for a flipping around the poles. (c) An embedding of the pertinent graph G_{R_1} compatible with $\Gamma^1_{R_1}$. Skeleton faces are drawn white and children faces are drawn black. Each skeleton face f is labeled with two integers representing its depths $d_l(f)$ and $d_r(f)$, respectively.

linear time by performing a single-source shortest path from the two external faces f^μ_l and f^μ_r [Tho99].

Given an R-node μ and one of its children ν, we propose an algorithm, called NESTED_RIGID, which, starting from $\hat{A}(\nu)$, $t(\mu)$, and the values of the depths $d_l(f)$ and $d_r(f)$ of each face f of Γ^1_μ, computes $S = \hat{A}(\nu|\Gamma^1_\mu)$.

The algorithm first generates a set S', containing a pair $\langle x_k + d_l(f^\nu_l), y_k + d_r(f^\nu_r)\rangle$ for each pair $\langle x_k, y_k\rangle \in \hat{A}(\nu)$, and a set S'', containing a pair $\langle y_k + d_l(f^\nu_r), x_k + d_r(f^\nu_l)\rangle$ for each pair $\langle x_k, y_k\rangle \in \hat{A}(\nu)$. Then, it initializes $S = S' \cup S''$.

For each pair $p^k = \langle x_k, y_k\rangle$ of $\hat{A}(\nu)$, define $p^k_{first} = \langle 0, \max(x_k + d_r(f^\nu_l), y_k + d_r(f^\nu_r))\rangle$. Denote by $\overline{p}_{first}$ the p^k_{first} with minimum y and by $\overline{p}_{last}$ the pair obtained from $\overline{p}_{first}$ by swapping the two elements x and y. If $\overline{p}_{first}$ is not preceded by the first pair of S, then insert $\overline{p}_{first}$ as the first element of S and append $\overline{p}_{last}$ to S as the last element. Remove from S any pair p^* such that either $\overline{p}_{first} \preceq p^*$ or $\overline{p}_{last} \preceq p^*$.

Property 5.9 *The set S computed by Algorithm* NESTED_RIGID *is succinct.*

Proof: S' is succinct and ordered with respect to the $\preceq_x$ relationship since it is obtained by adding the same constant values $d_l(f^\nu_l)$ and $d_r(f^\nu_r)$ to the first and second element, respectively, of all the pairs of $\hat{A}(\nu)$, which is succinct and assumed ordered with respect to the $\preceq_x$ relationship. For analogous

Algorithm 4 NESTED_RIGID

Require: An R-node μ, with its thickness $t(\mu)$ and the values of the depths $d_l(f)$ and $d_r(f)$ of each face of the embedding Γ^1_μ of $sk(\mu)$, and one of its children ν, with its gist $\hat{A}(\nu)$.

Ensure: The gist $\hat{A}(\nu|\Gamma^1_\mu)$ of the admissible set of ν nested into Γ^1_μ.

1: **for all** $p^k = \langle x_k, y_k \rangle \in \hat{A}(\nu)$ **do**
2: S'.addLast($\langle x_k + d_l(f_l^\nu), y_k + d_r(f_r^\nu) \rangle$);
3: S''.addLast($\langle y_k + d_l(f_r^\nu), x_k + d_r(f_l^\nu) \rangle$);
4: $p^k_{first} = \langle 0, \max(x_k + d_r(f_l^\nu), y_k + d_r(f_r^\nu)) \rangle$;
5: $p^k_{last} = \langle \max(x_k + d_l(f_l^\nu), y_k + d_l(f_r^\nu)), 0 \rangle$;
6: **if** $p^k_{first} \preceq \overline{p}_{first}$ **then**
7: $\overline{p}_{first} = p^k_{first}$;
8: **end if**
9: **end for**
10: $S = S' \cup S''$;
11: $p_{first} = S$.getFirst();
12: **if** $p_{first} \not\preceq \overline{p}_{first}$ **then**
13: S.addFirst($\overline{p}_{first}$);
14: $\overline{p}_{last} = \overline{p}_{first}$.swapElements();
15: S.addLast($\overline{p}_{last}$);
16: **for all** $p^* \neq \overline{p}_{first} \in S$ and $p^* \neq \overline{p}_{last} \in S$ **do**
17: **if** $\overline{p}_{first} \preceq p^*$ or $\overline{p}_{last} \preceq p^*$ **then**
18: S.remove(p^*);
19: **end if**
20: **end for**
21: **end if**
22: **return** S;

reasons also S'' is succinct and ordered with respect to the $\preceq_x$ relationship. By Lemma 5.4, the set $S' \cup S''$, computed with Algorithm GIST_UNION, is succinct. The statement follows from the fact that S is initialized to $S' \cup S''$ and, every time the two pairs $\overline{p}_{first}$ and $\overline{p}_{last}$ are added to S, none of its pairs precedes them and all of its pairs that are preceded by such pairs are removed from S. □

Property 5.10 *The set S computed by Algorithm* NESTED_RIGID *is a subset of $A(\nu|\Gamma^1_\mu)$.*

Proof: Consider a pair $p^k = \langle x_k, y_k\rangle \in \hat{A}(\nu)$. There exists an embedding Γ^k_ν such that the set of the internal faces of Γ^k_ν can be partitioned into two sets F^k_l and F^k_r such that all faces in F^k_l have distance from f^ν_l less or equal than x_k and all faces in F^k_r have distance from f^ν_r less or equal than y_k. Consider any embedding $\Gamma^*_{G_\mu}$ such that $\Gamma^*_{G_\mu}$ restricted to ν is Γ^k_ν.

Consider a pair $p \in S$. Four are the cases: $p \in S'$, $p \in S''$, $p = \overline{p}_{first}$, or $p = \overline{p}_{last}$. If $p \in S'$, then $p = \langle x_k + d_l(f^\nu_l), y_k + d_r(f^\nu_r)\rangle$, for some k. We show that $p \in \hat{A}(\nu|\Gamma^1_\mu)$ as follows. Consider the faces of the embedding $\Gamma^*_{G_\mu}$ internal to ν, that is, the faces corresponding to Γ^k_ν, and their partition into the sets F^k_r and F^k_l satisfying pair $\langle x_k, y_k\rangle$. For example, consider the partition of the internal faces of component R_2, shown in Fig. 5.12(a), satisfying pair $\langle 0, 1\rangle$. As shown in Fig. 5.12(b), faces $f \in F^k_l$ are at distance less or equal than $x_k + d_l(f^\nu_l)$ from f^μ_l, since they are at distance less or equal than x_k from f^ν_l which, in turn, is at distance $d_l(f^\nu_l)$ from f^μ_l. Faces $f \in F^k_r$ are at distance less or equal than $y_k + d_r(f^\nu_r)$ from f^μ_r, since they are at distance less or equal than y_k from f^ν_r which, in turn, is at distance $d_r(f^\nu_r)$ from f^μ_r. Analogous considerations, also shown in Fig. 5.12(c), prove that $p \in S''$ implies $p \in A(\nu|\Gamma^1_\mu)$.

If $p = \overline{p}_{first}$, then $p = \langle 0, max(x_k + d_r(f^\nu_l), y_k + d_r(f^\nu_r))\rangle$, for some k. We show that $p \in \hat{A}(\nu|\Gamma^1_\mu)$ as follows. Consider the faces of the embedding $\Gamma^*_{G_\mu}$ internal to ν. Faces in F^k_l are at distance less or equal than $x_k + d_r(f^\nu_l)$ from f^μ_r since they are at distance less or equal than x_k from f^ν_l which, in turn, is at distance $d_r(f^\nu_l)$ from f^μ_r. Faces in F^k_r are at distance less or equal than $y_k + d_r(f^\nu_r)$ from f^μ_r since they are at distance less or equal than y_k from f^ν_r which, in turn, is at distance $d_r(f^\nu_r)$ from f^μ_r. Therefore, each face is at distance less or equal than $max(x_k + d_r(f^\nu_l)), y_k + d_r(f^\nu_r))$ from f^μ_r, and hence $p \in \hat{A}(\nu|\Gamma^1_\mu)$. Analogous considerations prove that $p = \overline{p}_{last}$ implies $p \in A(\nu|\Gamma^1_\mu)$. □

Property 5.11 *The set S computed by Algorithm* NESTED_RIGID *precedes $A(\nu|\Gamma^1_\mu)$.*

Proof: Since $\hat{A}(\nu|\Gamma^1_\mu) \preceq A(\nu|\Gamma^1_\mu)$, it is sufficient to show that $S \preceq \hat{A}(\nu|\Gamma^1_\mu)$. Suppose, for a contradiction, that there exists a pair $p = \langle x, y\rangle \in \hat{A}(\nu|\Gamma^1_\mu)$ and there exists no pair $p' \in S$ such that $p' \preceq p$. Since $p \in \hat{A}(\nu|\Gamma^1_\mu)$, by definition there exists an embedding Γ_{G_μ}, compatible with Γ^1_μ, such that all faces of ν can

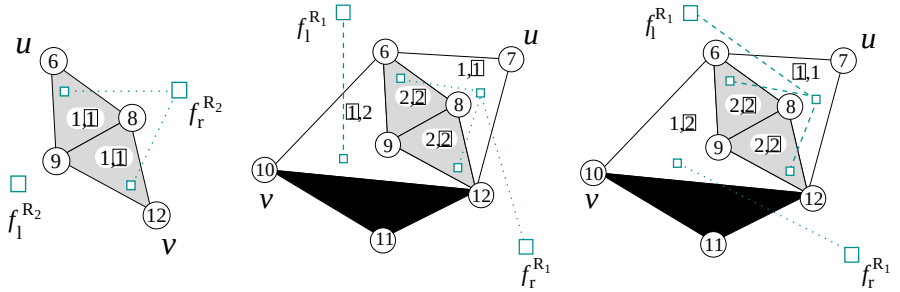


Figure 5.12: Nesting component R_2 into component R_1. (a) Component R_2 satisfies pair $p^k = \langle x_k, y_k \rangle = \langle 0, 1 \rangle$. (b) Component R_2 satisfies, nested into $\Gamma^1(R_1)$, pair $\langle x_k + d_l(f_l^\nu), y_k + d_r(f_r^\nu) \rangle = \langle 0+1, 1+1 \rangle = \langle 1, 2 \rangle$. (c) Component R_2 satisfies, nested into $\Gamma^1(R_1)$, pair $\langle y_k + d_l(f_r^\nu), x_k + d_r(f_l^\nu) \rangle = \langle 1+1, 0+2 \rangle = \langle 2, 2 \rangle$. Skeleton faces are drawn white, children faces not belonging to R_2 are drawn black, and light-shaded faces are in F_r^k. Each face is labeled with two integers representing its distance from the left and the right external faces, respectively. Labels lying inside a square are those used in the partition. Dashed edges are those used to reach the left external face, while dotted edges are those used to reach the right external face.

be partitioned into two sets F_l and F_r such that all faces in F_l are at distance from f_l^μ less or equal than x and all faces in F_r are at distance from f_r^μ less or equal than y.

Suppose that $x, y \neq 0$. Then, $F_l, F_r \neq \emptyset$. Consider the set P_l of the paths $\gamma(f, f_l^\mu)$ of minimum length from each face $f \in F_l$ to f_l^μ and the set P_r of the paths $\gamma(f', f_r^\mu)$ of minimum length from each face $f' \in F_r$ to f_r^μ. By the minimality of the paths and by the fact that $x, y \neq 0$, we may assume that all the paths in P_l contain f_l^ν and all the paths in P_r contain f_r^μ, or vice-versa. If this is not the case an equivalent partition $F_l', F_r' \neq \emptyset$ satisfying x, y and suitable minimum paths P_l' and P_r' can be found.

It follows that there exists a pair $p' = \langle x', y' \rangle \in A(\nu)$ such that either $x' = x - d_l(f_l^\nu)$ and $y' = y - d_r(f_r^\nu)$, or $x' = y - d_r(f_l^\nu)$ and $y' = x - d_l(f_r^\nu)$.

Consider the case $p' = \langle x - d_l(f_l^\nu), y - d_r(f_r^\nu) \rangle \in A(\nu)$. Denote by $\hat{p}'$ the pair $\hat{p}' = \langle \hat{x}', \hat{y}' \rangle \in \hat{A}(\nu)$ preceding p'. By the fact that for each pair $\langle x_k, y_k \rangle \in \hat{A}(\nu)$ Algorithm NESTED_RIGID adds a pair $p^k = \langle x_k + d_l(f_l^\nu), y_k + d_r(f_r^\nu) \rangle$ to S',

we have that there exists a pair $\hat{p}^k = \langle \hat{x}' + d_l(f_l^\nu), \hat{y}' + d_r(f_r^\nu)\rangle \in S'$ such that $\hat{p}^k \preceq \langle x - d_l(f_l^\nu) + d_l(f_l^\nu), y - d_r(f_r^\nu) + d_r(f_r^\nu)\rangle = \langle x, y\rangle = p$. Hence, by construction, either $\hat{p}^k \in S$ or there exists a pair $\overline{p} \in S$ such that $\overline{p} \preceq \hat{p}^k \preceq p$, contradicting the hypothesis that there is no pair in S preceding p.

Otherwise, if $p' = \langle y - d_r(f_l^\nu), x - d_l(f_r^\nu)\rangle$, analogous considerations show that there exists a pair $\overline{p} \in S$ that precedes a pair $\hat{p}^k \in S''$ such that $\hat{p}^k \preceq p$, contradicting the hypothesis that there is no pair in S preceding p.

Suppose $p = \langle 0, y\rangle$. Construct a partition of the internal faces of ν into two sets F_l' and F_r' such that $f \in F_r'$ if $f_r^\nu \in \gamma(f, f_r^\mu)$ and $f_l^\nu \notin \gamma(f, f_r^\mu)$, and $f \in F_l'$ in the other cases. In such a partition faces in F_r' are at distance less or equal than $y - d_r(f_r^\nu)$ from f_r^ν and faces in F_l' are at distance less or equal than $y - d_r(f_l^\nu)$ from f_l^ν. It follows that pair $p' = \langle y - d_r(f_l^\nu), y - d_r(f_r^\nu)\rangle$ belongs to $A(\nu)$. By the fact that Algorithm NESTED_RIGID builds a pair $p^k_{first} = \langle 0, \max(x_k + d_r(f_l^\nu), y_k + d_r(f_r^\nu))\rangle$ for each pair $p^k = \langle x_k, y_k\rangle$ of $\hat{A}(\nu)$ and possibly adds to S the pair, called $\overline{p}_{first}$, with minimum y among them, we have that there exists a pair $\hat{p}^k = \langle 0, \max(\hat{x}_k + d_r(f_l^\nu), \hat{y}_k + d_r(f_r^\nu))\rangle$ such that $\hat{p}^k \preceq \langle 0, \max(y - d_r(f_l^\nu) + d_r(f_l^\nu), y - d_r(f_r^\nu) + d_r(f_r^\nu)\rangle = \langle 0, \max(y, y)\rangle = \langle 0, y\rangle = p$. Hence, by construction, either $\hat{p}^k \in S$ or there exists a pair $\overline{p} \in S$ such that $\overline{p} \preceq \hat{p}^k \preceq p$, contradicting the hypothesis that there is no pair in S preceding p.

Analogous considerations show that, if $p = \langle x, 0\rangle$, then there exists a pair $\overline{p} \in S$ that precedes $\hat{p}^k \preceq p$, contradicting the hypothesis that there is no pair in S preceding p. □

Lemma 5.15 *Starting from $\hat{A}(\nu_i)$ and $t(\mu)$, Algorithm* NESTED_RIGID *computes $\hat{A}(\nu_i|\Gamma^1_\mu)$ in time $O(n(\nu_i))$.*

Proof: The set S computed by Algorithm NESTED_RIGID is such that $S \preceq A(\nu_i|\Gamma^1_\mu)$ (Property 5.11) and $S \subseteq A(\nu_i|\Gamma^1_\mu)$ (Property 5.10). It follows that S reduces $A(\nu_i|\Gamma^1_\mu)$. Further, by Property 5.9, S is succinct. Hence, S is $\hat{A}(\nu_i|\Gamma^1_\mu)$. Now we show that Algorithm NESTED_RIGID runs in $O(n(\nu_i))$ time. The construction of sets S' and S'' can be performed in $O(n(\nu_i))$ time, since a constant number of operations is performed for each pair of $\hat{A}(\nu_i)$. Lemma 5.4 ensures that the union of S' and S'' is computed by Algorithm GIST_UNION in $O(n(\nu_i))$ time, since $|S'| = |S''| = O(n(\nu_i))$. The construction of the pairs $\overline{p}_{first}$ and $\overline{p}_{last}$, as well as their insertion into S, is performed in $O(n(\nu_i))$ time, since in both cases each element of the set is considered only once. □

By using the above discussed algorithm, for each child ν_i of μ, we compute the set $\hat{A}(\nu_i|\Gamma^1_\mu)$. As described in the beginning of this section, in order to find

$\hat{A}^1(\mu)$, we need to compute also $\hat{A}(\Gamma^1_\mu)$, which is the gist of the admissible set of the skeleton of μ restricted to its embedding Γ^1_μ. We propose an algorithm, called SKELETON_RIGID, which, starting from the values of the depths $d_l(f)$ and $d_r(f)$ of each face f of Γ^1_μ, computes $S = \hat{A}(\Gamma^1_\mu)$.

Set S is initialized to $\emptyset$. Let $x^{max} = \max_{f \in \Gamma^1_\mu}\{d_l(f)\}$. Then, for increasing values of x, starting from zero and ending with x^{max}, construct a partition of the internal faces of Γ^1_μ into the two sets F_l and F_r such that all faces at distance less or equal than x from f^μ_l are in F_l and all the other faces are in F_r. Compute the maximum distance y from the faces in F_r to f^μ_r. If $\langle x, y\rangle$ is incomparable with the last pair added to S, add $\langle x, y\rangle$ to S.

Algorithm 5 SKELETON_RIGID

Require: An R-node μ, with the values of the depths $d_l(f)$ and $d_r(f)$ of each internal face f of the embedding Γ^1_μ of $sk(\mu)$.

Ensure: The gist $\hat{A}(\Gamma^1_\mu)$ of the admissible set of $sk(\mu)$.

1: $S = \emptyset$;
2: **for** $x = 0$ to $\max_{f \in \Gamma^1_\mu}\{d_l(f)\}$ **do**
3: $\quad p = \langle x, \max_{f \in \Gamma^1_\mu \ \&\ d_l(f) > x}\{d_r(f)\}\ \rangle$;
4: $\quad$ **if** S.getLast() $\nsim p$ **then**
5: $\quad\quad S$.addLast(p);
6: $\quad$ **end if**
7: **end for**
8: **return** S;

Lemma 5.16 *Starting from the values of the depths $d_l(f)$ and $d_r(f)$ of each internal face f of Γ^1_μ, Algorithm* SKELETON_RIGID *computes the gist $\hat{A}(\Gamma^1_\mu)$ of the admissible set of $sk(\mu)$ in $O(\delta(\mu))$ time.*

Proof: Let S be the set of integer pairs computed by Algorithm SKELETON_RIGID. We prove the statement by proving that (a) S is succinct; (b) $S \subset A(\Gamma^1_\mu)$; and (c) $S \preceq A(\Gamma^1_\mu)$.

(a) (S is succinct) We prove that pair p_i added to S by Algorithm SKELETON_RIGID at step i is incomparable with pairs $p_1, p_2, \ldots, p_{i-1}$ added to S at the previous steps. For $i = 1$ the statement is trivially true.

Consider step $i > 1$. By Property 5.1.5 and by the fact that pairs $\langle x, y\rangle$ are considered for increasing values of x, we have that if $p_i \nsim p_{i-1}$, then $p_i \nsim p_j$, for $j = 1, \ldots, i-1$. Hence, it is sufficient to prove that $p_i = \langle x_i, y_i\rangle \nsim p_{i-1} = \langle x_{i-1}, y_{i-1}\rangle$. Since, by construction, p_i is inserted only if it is incomparable with the last pair of S, we have that S is succinct.

(b) ($S \subset A(\Gamma^1_\mu)$) Each pair $p \in S$ is built considering a particular partition of the internal faces of Γ^1_μ into the two sets F_l and F_r which satisfies p. Hence, by definition, $p \in A(\Gamma^1_\mu)$.

(c) ($S \preceq A(\Gamma^1_\mu)$) Suppose, for a contradiction, that there exists a pair $p' = \langle x', y'\rangle \in A(\Gamma^1_\mu)$ and there exists no pair $p \in S$ such that $p \preceq p'$. Let $x^{max} = \max_{f \in \Gamma^1_\mu}\{d_l(f)\}$. Two are the cases: either $x' > x^{max}$ or not.

If $x' > x^{max}$, consider the pair $p = \langle x, y\rangle$ created by Algorithm SKELETON_RIGID such that $x = x^{max}$. By construction, it is possible to find a partition of the internal faces of Γ^1_μ into two sets F_l and F_r such that all faces are in F_l and $F_r = \emptyset$. Hence, we have that $y = 0$. By the fact that $x < x'$ and $0 \leq y'$, we have that $p \preceq p'$. Since, by construction, either $p \in S$ or there exists a pair $p'' \in S$ such that $p'' \preceq p \preceq p'$, we have a contradiction to the hypothesis that there exists no pair $p \in S$ such that $p \preceq p'$.

Otherwise, if $x' \leq x^{max}$, consider the pair $p = \langle x, y\rangle$ created by Algorithm SKELETON_RIGID such that $x = x'$ and consider the two partitions F_l, F_r and F'_l, F'_r satisfying p and p', respectively. Since, by construction, all the faces assigned to F_r by Algorithm SKELETON_RIGID are at distance greater than x from f^μ_l, we have that $F_r \subseteq F'_r$. Hence, $y \leq y'$ and, since $x = x'$, we have that $p \preceq p'$. Since, by construction, we have that either $p \in S$ or there exists a pair $p'' \in S$ such that $p'' \preceq p \preceq p'$, we have a contradiction to the hypothesis that there exists no pair $p \in S$ such that $p \preceq p'$.

The time complexity bound follows from the fact that Algorithm SKELETON_RIGID iterates $x^{max} = \max_{f \in \Gamma^1_\mu}\{d_l(f)\}$ times, which, by Property 5.5, is $O(\delta(\mu))$, and each iteration is executed in $O(1)$ time. □

Lemma 5.17 *Starting from $\hat{A}(\nu_i)$ and $t(\nu_i)$, for $i = 1, \ldots, \delta(\mu)$, the gist $\hat{A}(\mu)$ can be computed in time $O(\sum_{j=1}^{\delta(\mu)} \sum_{i=1}^{j} n(\nu_i))$.*

Proof: In order to compute $\hat{A}(\mu)$ we have to perform a sequence of computations. The proof is based on the fact that the heaviest of these computations needs $O(\sum_{j=1}^{\delta(\mu)} \sum_{i=1}^{j} n(\nu_i))$ time. By Lemma 5.14, the thickness $t(\mu)$ can be computed in $O(\delta(\mu))$ time. Given the embedding Γ_μ^1 of $sk(\mu)$, by Lemma 5.16 the gist $\hat{A}(\Gamma_\mu^1)$ can be computed in $O(\delta(\mu))$ time. By Lemma 5.15, the gists $\hat{A}(\nu_i|\Gamma_\mu^1)$ of the admissible sets of ν_i nested into Γ_μ^1 can be computed in $O(\sum_{i=1}^{\delta(\mu)} n(\nu_i))$ total time. By Lemma 5.9, the gist $\hat{A}^1(\mu)$ of the admissible set $A^1(\mu) = (\bigcap_{i=1}^{\delta(\mu)} A(\nu_i|\Gamma_\mu^1)) \cap A(\Gamma_\mu^1)$ of component μ restricted to the embedding Γ_μ^1 can be computed in $O(\sum_{j=1}^{\delta(\mu)} \sum_{i=1}^{j} n(\nu_i))$ time. The gist $\hat{A}^2(\mu)$ of the admissible set of μ restricted to the second embedding Γ_μ^2 of $sk(\mu)$ can be computed in $O(\sum_{i=1}^{\delta(\mu)} n(\nu_i))$ time by adding to $\hat{A}^2(\mu)$ a pair $\langle y, x\rangle$ for each pair $\langle x, y\rangle \in \hat{A}^1(\mu)$. The gist $\hat{A}(\mu)$ of the admissible set $A(\mu) = A^1(\mu) \cup A^2(\mu)$ is computed in $O(\sum_{i=1}^{\delta(\mu)} n(\nu_i))$ time, according to Lemma 5.4. □

The Parallel Case.

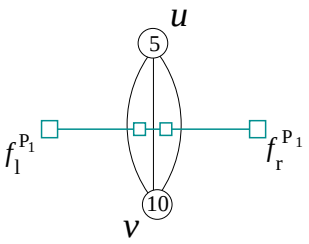


Figure 5.13: One of the $\delta(\mu)!$ embeddings of the skeleton $sk(P_1)$ of the parallel component P_1 and the corresponding (u, v)-dual.

Let μ be a P-node with children ν_i, for $i = 1, \ldots, \delta(\mu)$, and let $n(\nu_i)$ be the number of vertices of ν_i. The computation of the thickness is addressed by the following lemma.

Lemma 5.18 *The thickness $t(\mu) = \sum_{i=1}^{\delta(\mu)} t(\nu_i)$ can be computed in $O(\delta(\mu))$ time.*

Proof: The (u,v)-dual graph of each embedding of $sk(\mu)$ is made up of a single path of length $\delta(\mu)$ connecting the two vertices corresponding to f_l^μ and f_r^μ and passing for all the vertices corresponding to the internal faces (see, for example, Fig. 5.13). Since, by definition, $t(\mu)$ is the distance between f_l^μ and f_r^μ in such a (u,v)-dual graph, the statement follows. □

The remaining part of this section is devoted to the computation of the admissible set of μ. In the parallel case the skeleton $sk(\mu)$ of a component μ is composed of two vertices, the poles u and v, with $\delta(\mu)$ parallel edges connecting them and admits a factorial number of embeddings, that is, the number of all possible permutations of its $\delta(\mu)$ edges. In Fig. 5.13 one of the embeddings of $sk(\mu)$ is shown together with its corresponding (u,v)-dual. Hence, according to Lemma 5.10, the gist $\hat{A}(\mu)$ can be obtained by performing the union among $\delta(\mu)!$ sets, where each set $A^j(\mu)$ corresponds to a different embedding Γ_μ^j of $sk(\mu)$. Also, $A^j(\mu)$ can be computed by intersecting $\hat{A}(\Gamma_\mu^j)$ and the gists $\hat{A}(\nu_i|\Gamma_\mu^j)$ of the admissible sets of each component ν_i nested into Γ_μ^j. Hence, a naïve computation of $\hat{A}(\mu)$ employs a factorial number of steps. We reduce the number of permutations to be analyzed by exploiting the following properties and considerations.

Let μ be a P-node with children ν_i, for $i = 1, \ldots, \delta(\mu)$. First, consider a pair $\langle x, y\rangle \in \hat{A}(\mu)$ and an embedding Γ_{G_μ} of G_μ satisfying $\langle x, y\rangle$, i.e., whose internal faces can be partitioned into two sets F_l and F_r such that all faces in F_l (F_r) have distance from f_l^μ (f_r^μ) less or equal than x (y). Given a face $f \in \Gamma_{G_\mu}$, consider paths $\gamma(f, f_l^\mu)$ of minimum length from f to f_l^μ, and paths $\gamma(f, f_r^\mu)$ of minimum length from f to f_r^μ. The following properties hold.

Property 5.12 *Let ν be one of the child components of μ. If the children faces of Γ_{G_μ} corresponding to the internal faces of ν are split by F_l and F_r, then, for each face $f \in F_l$, we have that $f_l^\nu \in \gamma(f, f_l^\mu)$ and $f_r^\nu \notin \gamma(f_l^\nu, f_l^\mu)$. Analogously, for each face $f \in F_r$, we have that $f_r^\nu \in \gamma(f, f_r^\mu)$ and $f_l^\nu \notin \gamma(f_r^\nu, f_r^\mu)$.*

Proof: Consider the sequence of skeleton faces $f_l^\mu = f_1, f_2, \ldots, f_{\delta(\mu)}, f_{\delta(\mu)+1} = f_r^\mu$. The values of left depths $d_l(f_i)$, with $i = 1, \ldots, \delta(\mu)+1$, are increasing, while the values of the depths $d_r(f_i)$ are decreasing. Hence, $d_l(f_l^\nu) < d_l(f_r^\nu)$ and $d_r(f_l^\nu) > d_r(f_r^\nu)$. Consider a face $f \in F_l$. Since $\gamma(f, f_l^\mu)$ is a path of minimum length and $d_l(f_l^\nu) < d_l(f_r^\nu)$, the statement follows. The same for a face $f \in F_r$. □

Property 5.13 *Let f_{sk} be a skeleton face of Γ_{G_μ} and let ν_l and ν_r be the two child components of μ incident to f_{sk} such that $f_r^{\nu_l} = f_{sk} = f_l^{\nu_r}$. If some*

internal face f_l of ν_l belongs to F_r and some internal face f_r of ν_r belongs to F_l, then there exists a partition F'_l, F'_r such that all faces in F'_l have distance from f_l^μ less or equal than x and all faces in F'_r have distance from f_r^μ less or equal than y and such that either all internal faces of ν_l belong to F_l or all internal faces of ν_r belong to F_r.

Proof: By Property 5.12, we have that, for each face f_l internal to ν_l, $f_{sk} = f_r^{\nu_l} \in \gamma(f_l, f_l^\mu)$ and, for each face f_r internal to ν_r, $f_{sk} = f_l^{\nu_r} \in \gamma(f_r, f_l^\mu)$.

Denote by $d'_l(\nu_r)$ the maximum distance from $f_l^{\nu_r} = f_{sk}$ of the internal faces of ν_r belonging to F_l, and by $d'_r(\nu_l)$ the maximum distance from $f_r^{\nu_l} = f_{sk}$ of the internal faces of ν_l belonging to F_r.

If $d'_l(\nu_r) < d'_r(\nu_l)$, then consider the partition F'_l, F'_r obtained from F_l and F_r by moving all the internal faces of ν_r to F_r. By $F'_l \subset F_l$, we have that all faces in F'_l are at distance from f_l^μ less or equal than x; by construction, faces $f \in F_r$ are at distance from f_r^μ less or equal than y; also, by construction, each face f_r internal to ν_r is at distance from f_r^μ less or equal than $d'_l(\nu_r) + d_r(f_{sk}) < d'_r(\nu_l) + d_r(f_{sk}) \leq y$. Hence, partition F'_l, F'_r satisfies the conditions of the statement.

Analogously, if $d'_l(\nu_r) > d'_r(\nu_l)$, then the partition F'_l, F'_r obtained from F_l and F_r by moving all the internal faces of ν_l to F_l is such that all faces in F'_l have distance from f_l^μ less or equal than x and all faces in F'_r have distance from f_r^μ less or equal than y. Fig. 5.14 shows the latter case ($d'_l(\nu_r) > d'_r(\nu_l)$). □

Lemma 5.19 *Let Γ_{G_μ} be an embedding of G_μ satisfying pair $\langle x, y \rangle \in \hat{A}(\mu)$. There exists a partition F'_l and F'_r such that:*

1. *All faces in F'_l have distance from f_l^μ less or equal than x;*
2. *all faces in F'_r have distance from f_r^μ less or equal than y;*
3. *there exists at most one component ν_c whose internal faces belong to both the sets F'_l and F'_r;*
4. *each child component to the left of ν_c has its internal faces in F'_l and each child component to the right of ν_c has its internal faces in F'_r.*

Proof: If there exist in Γ_{G_μ} two child components ν_1 and ν_2, possibly not sharing an external face, with ν_1 to the left of ν_2, and such that some of the internal faces of ν_1 are in F_r and some of the internal faces of ν_2 are in F_l, then

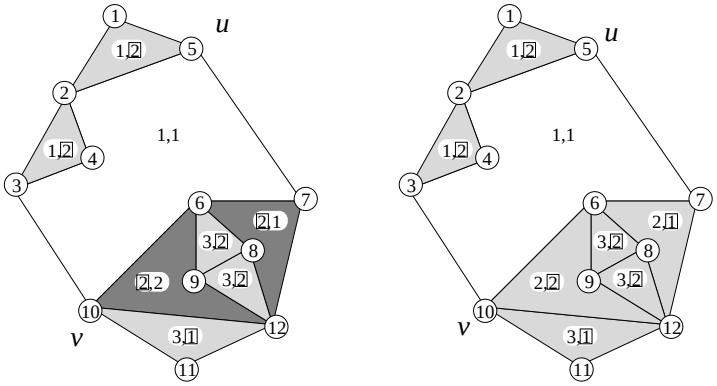
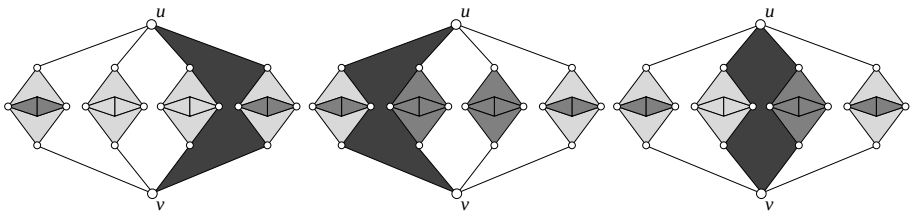


Figure 5.14: Property 5.13. Component S_1 is ν_l and component S_2 is ν_r. (a) Parallel component $\{S_1, S_2\}$ satisfies pair $\langle 2, 2\rangle$ when some faces of S_1 are in F_r and some faces of S_2 are in F_l. (b) Parallel component $\{S_1, S_2\}$ satisfies pair $\langle 0, 2\rangle \preceq \langle 2, 2\rangle$ when all faces of S_2 are in F'_r. Dark-shaded faces are in F_l, light-shaded faces are in F_r and F'_r, and face f_{sk} is drawn white. Each face is labeled with two integers representing its distance from left and right external face, respectively. Labels lying inside a square are those considered in the corresponding partition.

there is at least one skeleton face f_{sk} whose two incidents components ν_l and ν_r, with $f_r^{\nu_l} = f_{sk} = f_l^{\nu_r}$, have some internal face in F_r and some internal face in F_l, respectively. A proof of this fact is shown in Fig. 5.15, where the first component ν_1 has some faces in F_r and the last component ν_2 has some faces in F_l. Observe that we analyze only the case where components lying between ν_1 and ν_2 have all their internal faces in the same set since, in the other case, it is possible to find at least one subsequence of components where the above property is satisfied.

By Property 5.13, it is possible to find a partition F'_l, F'_r such that all faces in F'_l have distance from f_l^{μ} less or equal than x, all faces in F'_r have distance from f_r^{μ} less or equal than y and either all internal faces of ν_l belong to F'_l or all internal faces of ν_r belong to F'_r. Partition F'_l, F'_r can be repeatedly modified as described above until Property 5.13 can not be further applied. Two are the cases: either there is a single child component ν_c whose internal



(a) (b) (c)

Figure 5.15: Proof of Lemma 5.19. The leftmost component is ν_1 and the rightmost component is ν_2. Dark-shaded faces are in F_l, light-shaded faces are in F_r, f_{sk} is drawn dark grey, while the other skeleton faces are drawn white.

faces belong to both the sets F'_l and F'_r or there is none. In the first case, since Property 5.13 can not be applied, all child components to the left (right) of ν_c have their internal faces in F'_l (F'_r, respectively). In the second case, for analogous reasons, all child components with their faces in F'_l are to the left of those child components whose faces are in F'_r. □

The unique component ν_c, if any, whose faces belong to both F'_l and F'_r is called the *center* of the permutation. Intuitively, Lemma 5.19 states that we can restrict to consider those partitions F_l, F_r of the internal faces of Γ_{G_μ} such that each child component different from ν_c has its internal faces into the same set F_l or F_r. Consider, for example, embedding $\Gamma^j_{P_1}$ of parallel component P_1, shown in Fig. 5.16, corresponding to permutation $S_1, Q_{5,10}, S_2$, and consider $Q_{5,10}$ as the center of the permutation. It is possible to observe that a partition F_l, F_r, where the children faces of S_2 are split by the two sets (Fig. 5.16(a)), satisfies a pair that is preceded by a pair satisfied by a partition where all faces of S_2 are in F_r(Fig. 5.16(b)). In other words, for each child component ν_i different from ν_c, $\hat{A}(\nu_i)$ can be assumed to contain the two pairs $\langle x, 0\rangle$ and $\langle 0, y\rangle$ only.

The gist $\hat{A}(\mu)$ can be computed by choosing, one by one, each child component as the center of the permutation ν_c and by inserting the other components either to the left or the right of ν_c, until a complete permutation is obtained. Each subsequence σ of components is associated with the gist $\hat{A}(\sigma)$ of its admissible set $A(\sigma)$, which is properly updated when a component is inserted. This approach would obtain the same permutation $\delta(\mu)$ times, so exploring $O(\delta(\mu) \cdot \delta(\mu)!)$ sequences. Hence, at first glance, the computational complex-

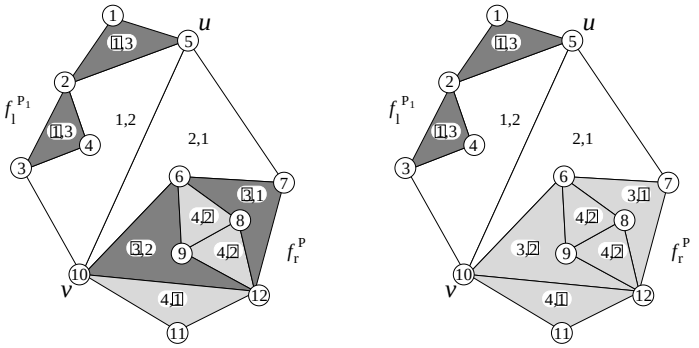


Figure 5.16: Embedding $\Gamma^j_{P_1}$ corresponds to permutation $S_1, Q_{5,10}, S_2$ and $Q_{5,10}$ is the center of the permutation. (a) $\Gamma^j_{P_1}$ satisfies pair $\langle 3,2\rangle$ when children faces of S_2 are split by F_l and F_r. (b) $\Gamma^j_{P_1}$ satisfies pair $\langle 1,2\rangle \preceq \langle 3,2\rangle$ when all children faces of S_2 are in F_r. Skeleton faces are drawn white, dark-shaded faces are in F_l, and light-shaded faces are in F_r. Each face is labeled with two integers representing its distance from left and right external face, respectively. Labels lying inside a square are those considered in the corresponding partition.

ity is augmented. However, we show in the following that focusing on ν_c can greatly help to reduce the number of permutations to be considered.

Lemma 5.20 *Let σ be a sequence of child components, with $\nu_c \in \sigma$, and let $\nu_i \notin \sigma$ be a child component of μ with $\langle 0, y_i\rangle, \langle x_i, 0\rangle \in \hat{A}(\nu_i)$. By adding ν_i to the left of σ we obtain a sequence σ'. Consider the set S' containing a pair $\langle \max(x + t(\nu_i), x_i), y\rangle$ for each pair $\langle x, y\rangle \in \hat{A}(\sigma)$. We have that $S' \preceq A(\sigma')$ and $S' \subset A(\sigma')$. Analogously, by adding ν_i to the right of σ we obtain a sequence σ'' and the set S'' containing a pair $\langle x, \max(y + t(\nu_i), y_i)\rangle$ for each pair $\langle x, y\rangle \in \hat{A}(\sigma)$ is such that $S'' \preceq A(\sigma'')$ and $S'' \subset A(\sigma'')$.*

Proof: First, we show that $S' \subset A(\sigma')$. Consider a pair $\langle x, y\rangle \in \hat{A}(\sigma)$ and the embedding Γ_σ of σ such that there exists a partition of its internal faces into two sets F_l and F_r where faces in F_l are at distance from f^σ_l less or equal than x and faces in F_r are at distance from f^σ_r less or equal than y. Then, consider the embedding Γ_{ν_i} of ν_i such that all its internal faces are at distance from $f^{\nu_i}_l$

less or equal than x_i and the planar embedding $\Gamma_{\sigma'}$ of σ' obtained by adding ν_i to the left of σ while preserving the original embeddings of σ and ν_i. Consider the partition of the internal faces of $\Gamma_{\sigma'}$ into two sets F'_l and F'_r, where F'_l is the set of faces composed of F_l plus all the faces of Γ_{ν_i} and $F'_r = F_r$. Faces in F_l are at distance less or equal than $x + t(\nu_i)$ from $f_l^{\sigma'}$, since they are at distance less or equal than x from f_l^{σ} which, in turn, is at distance $t(\nu_i)$ from $f_l^{\sigma'}$. Also, by construction, faces in Γ_{ν_i} are at distance less or equal than x_i from $f_l^{\sigma'}$. Hence, all faces in F'_l are at distance from $f_l^{\sigma'}$ less or equal than $\max(x + t(\nu_i), x_i)$. By $F'_r = F_r$ we have that $\langle \max(x + t(\nu_i), x_i), y\rangle \in A(\sigma')$. Repeating such a procedure for each pair $\langle x, y\rangle \in \hat{A}(\sigma)$ we have that $S' \subset A(\sigma')$.

Second, we show that $S' \preceq A(\sigma')$. Suppose, for a contradiction, that there exists a pair $p' = \langle x', y'\rangle$ such that $p' \in A(\sigma')$ and there is not a pair $p \in S'$ such that $p \preceq p'$. Since $p' \in A(\sigma')$, by definition, there exists an embedding $\Gamma_{\sigma'}$ whose internal faces can be partitioned into two sets F'_l and F'_r such that all faces in F'_l are at distance from $f_l^{\sigma'}$ less or equal than x' and all faces in F'_r are at distance from $f_r^{\sigma'}$ less or equal than y'. Consider the embeddings Γ_σ of σ and Γ_{ν_i} of ν_i induced by $\Gamma_{\sigma'}$. By Property 5.13, since $\nu_c \in \sigma$ and ν_i is to the left of σ, we have that Γ_{ν_i} has all its internal faces into F'_l. Hence, $x_i \leq x'$. Consider the partition of the internal faces of Γ_σ into the sets $F_l = F'_l \backslash \Gamma_{\nu_i}$ and $F_r = F'_r$. We have that faces $f_l \in F_l$ are at distance less or equal than $x' - t(\nu_i)$ from f_l^{σ}, since $f_l^{\sigma} \in \gamma(f_l, f_l^{\sigma'})$ and $|\gamma(f_l^{\sigma}, f_l^{\sigma'})| = t(\nu_i)$, and faces in F_r are at distance y' from f_r^{σ}, since $F_r = F'_r$. Hence, embedding Γ_σ of σ satisfies pair $\langle x' - t(\nu_i), y'\rangle$. Hence, by construction, S' contains a pair $p = \langle \max(x + t(\nu_i), x_i), y\rangle = \langle \max(x' - t(\nu_i) + t(\nu_i), x_i), y'\rangle = \langle \max(x', x_i), y'\rangle = \langle x', y'\rangle \preceq p'$, which is a contradiction.

Analogous considerations show that, when adding ν_i to the right of σ, $S'' \preceq A(\sigma'')$ and $S'' \subset A(\sigma'')$. □

Let ν_i be a child component of a parallel node. We introduce function $l(\nu_i) = t(\nu_i) - x_i$, where x_i is such that pair $\langle x_i, 0\rangle \in \hat{A}(\nu_i)$. The following lemma holds.

Lemma 5.21 *Let σ be a sequence of child components of μ, with $\nu_c \in \sigma$, and let ν' and ν'' be two child components of μ, with $\nu', \nu'' \notin \sigma$. Let Γ_μ^1 and Γ_μ^2 be two embeddings of $sk(\mu)$ corresponding to two permutations of child components which only differ for the swapping of ν' and ν'' in such a way that ν' and ν'' lie on the same side of σ and that ν' is between ν_c and ν'' in Γ_μ^1. If $l(\nu') < l(\nu'')$, then $A^2(\mu) \preceq A^1(\mu)$.*

Proof: Consider a pair $\langle x, y\rangle \in \hat{A}(\sigma)$, and pairs $\langle x', 0\rangle \in \hat{A}(\nu')$ and $\langle x'', 0\rangle \in$

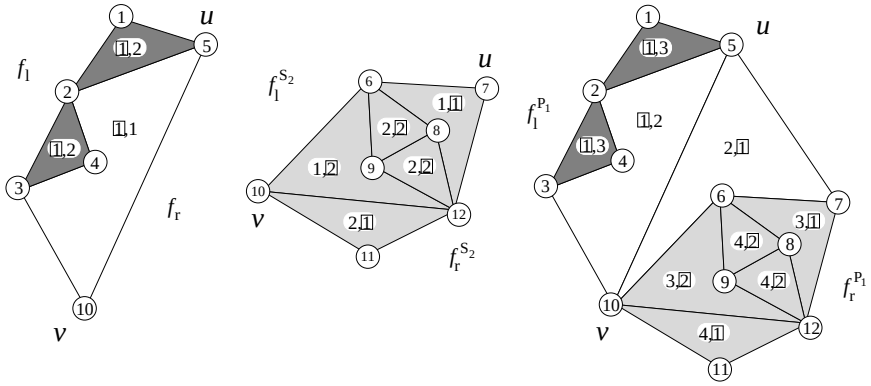


Figure 5.17: (a) Parallel component $\{S_1, Q_{5,10}\}$ satisfies pair $\langle 1, 0\rangle$. (b) Pair $\langle 0, 2\rangle \in \hat{A}(S_2)$. (c) After inserting S_2 to the right of the center, parallel component $P_1 = \{S_1, Q_{5,10}, S_2\}$ satisfies pair $\langle x_j, \max(y_j + t(\nu_i), y_i)\rangle = \langle 1, \max(0+1, 2)\rangle = \langle 1, 2\rangle$. Skeleton faces are drawn white, dark-shaded faces are in F_l, and light-shaded faces are in F_r. Each face is labeled with two integers representing its distance from left and right external face, respectively. Labels lying inside a square are those considered in the corresponding partition.

$\hat{A}(\nu'')$. We analyze the case when both ν' and ν'' are added to the left of σ, the other case being analogous. First, consider embedding Γ^1_μ. When component ν' is added to the left of σ, by Lemma 5.20, we obtain a new sequence satisfying pair $\langle \max(x+t(\nu'), x'), y\rangle$. Then, when ν'' is inserted, we obtain a new sequence satisfying $p^1 = \langle \max(\max(x+t(\nu'), x')+t(\nu''), x''), y\rangle$. Second, consider embedding Γ^2_μ. In this case, after inserting ν'' and ν', we obtain a new sequence satisfying pair $p^2 = \langle \max(\max(x + t(\nu''), x'') + t(\nu'), x'), y\rangle$. Suppose, for a contradiction, that $p^2 \not\preceq p^1$. Two are the cases: either $x + t(\nu') > x'$ or not. Consider the case $x + t(\nu') > x'$. Since, by hypothesis, $t(\nu') - x' < t(\nu'') - x''$, we have $x+t(\nu'') > x''$. Hence, $p^1 = \langle \max(x+t(\nu')+t(\nu''), x''), y\rangle = \langle x+t(\nu')+t(\nu''), y\rangle$ and $p^2 = \langle \max(x+t(\nu'')+t(\nu'), x'), y\rangle = \langle x+t(\nu'')+t(\nu'), y\rangle = p^1$, contradicting the hypothesis that $p^2 \not\preceq p^1$. Now consider the case $x + t(\nu') \leq x'$. Then, $p^1 = \langle \max(x' + t(\nu''), x''), y\rangle$. Again, we have to distinguish two cases: Either $x+t(\nu'') > x''$ or not. If $x+t(\nu'') > x''$, then $p^2 = \langle \max(x+t(\nu'')+t(\nu'), x'), y\rangle$. Since, by hypothesis, $x' + t(\nu'') \geq x + t(\nu'') + t(\nu')$ and, by $t(\nu'') \geq 0$, it holds

$x' + t(\nu'') \geq x'$, we conclude that $p^2 \preceq p^1$, contradicting the hypothesis. If $x + t(\nu'') \leq x''$, then $p^2 = \langle \max(x'' + t(\nu'), x'), y \rangle$. Since, by $l(\nu') < l(\nu'')$, we have $x' + t(\nu'') > x'' + t(\nu')$ and, by $t(\nu'') \geq 0$, we have $x' + t(\nu'') > x'$, we conclude that $p^2 \preceq p^1$, contradicting the hypothesis. Since, for every pair $p^1 \in A^1(\mu)$, there exists a pair $p^2 \in A^2(\mu)$ such that $p^2 \preceq p^1$, it follows that $A^2(\mu) \preceq A^1(\mu)$. □

Let ν' and ν'' be two child components of μ with $l(\nu'') < l(\nu')$. Intuitively, any permutation of the child components of μ with ν' further from ν_c than ν'' can be ignored, since its admissible set is preceded by the one computed with a different permutation. Therefore, the number of analyzed permutations can be reduced by ordering child components by decreasing values of function $l(\nu_i)$ and, once the center ν_c of the permutation has been chosen, by adding the other components, ordered with respect to function $l(\nu_i)$, either to the left or to the right of the sequence of the components already added.

In order to do so, we build a rooted tree $T(\nu_c)$ of height $\delta(\mu)+1$ where each node p at distance d from the root is a pair $\langle x_p, y_p \rangle$ of non-negative integers and is associated with a sequence σ_p of d child components of μ such that $\langle x_p, y_p \rangle \in \hat{A}(\sigma_p)$. The nodes at distance d from the root are the incomparable pairs of integers satisfied by some sequence of length d. Hence, the set of nodes at distance $\delta(\mu)$ from the root is $\hat{A}(\mu)$ restricted to the permutations having ν_c as the center. Such a set is denoted by $\hat{A}_{\nu_c}(\mu)$. Tree $T(\nu_c)$ is built as follows. The root is pair $\langle 0, 0 \rangle$ and is associated with the empty sequence. The root is added as many children as many pairs in the gist $\hat{A}(\nu_c)$ of the admissible set of the center of the permutation ν_c, each one associated with the sequence composed by ν_c only. The following levels are obtained by considering, one by one, all the other components in decreasing order of function $l(\nu_i)$. When the k-th component ν_k is processed, each node p at depth $k-1$ is added two children p_l and p_r, corresponding to the addition of ν_k to the left or to the right of σ_p, respectively. Pairs p_l and p_r are computed, starting from p, with the function described in Lemma 5.20. From the set of all pairs introduced at level k all those preceded by a pair of the same level can be removed, so pruning the tree.

The gist $\hat{A}(\mu)$ of the admissible set of the P-node μ can be obtained as the union of the gists $\hat{A}_{\nu_c}(\mu)$ of the admissible sets obtained by choosing, one by one, each child component as the center of the permutation ν_c.

Consider, for example, the P-node P_1 with child components $Q_{5,10}$, S_1, and S_2 shown in Fig. 5.17. Regarding component $Q_{5,10}$ we have $\langle 0, 0 \rangle \in \hat{A}(Q_{5,10})$ and $t(Q_{5,10}) = 1$, regarding component S_1 we have $\langle 1, 0 \rangle \in \hat{A}(S_1)$ and $t(S_1) =$

1, and regarding component S_2 we have $\langle 0,2\rangle \in \hat{A}(S_2)$ and $t(S_2) = 1$. Hence, $l(Q_{5,10}) = 1$, $l(S_1) = 0$ and $l(S_2) = -1$. The ordering of the components is then $Q_{5,10} - S_1 - S_2$. Consider $Q_{5,10}$ as the center of the permutation. We have $\hat{A}(Q_{5,10}) = \{\langle 0,0\rangle\}$. The tree that computes the admissible set restricted to all the permutations having $Q_{5,10}$ as center is shown in Fig. 5.18.

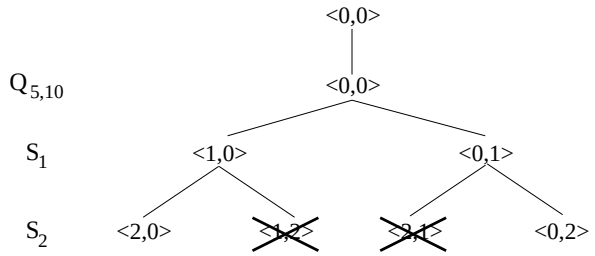


Figure 5.18: Computation of tree $T(Q_{5,10})$ for the P-node P_1 with child components $Q_{5,10}$, S_1, and S_2 added in this order. Pairs $\langle 1,2\rangle$ and $\langle 2,1\rangle$ have been removed since they are preceded by $\langle 0,2\rangle$ and $\langle 2,0\rangle$, respectively.

We propose an algorithm, called TREE_PARALLEL, which, given a P-node μ and its children ν_i, for $i = 1, \ldots, \delta(\mu)$, with one of them chosen as the center of the permutation ν_c, suitably builds a set S that is the gist $\hat{A}_{\nu_c}(\mu)$ of the admissible set restricted to all the permutations having ν_c as center. Components ν_i are considered ordered with respect to decreasing values of function $l(\nu_i)$. At each step of the computation, S contains the pairs associated to the nodes of the current level of the tree $T(\nu_c)$. S is initialized to $\hat{A}(\nu_c)$. When adding the k-th component ν_k, first produce the two sets S_l and S_r of children obtained by concatenating ν_k to the left and to the right of σ_p, respectively, for each $p \in S$. Pairs in S_l and S_r are created by applying to each pair of S the function presented in Lemma 5.20. Since S is ordered with respect to the $\preceq_x$ relationship, and due to the fact that pairs in S_l and S_r are created by adding a constant value to the first and second element, respectively, of all the pairs of S, which is succinct and assumed ordered with respect to the $\preceq_x$ relationship, these two sets can be kept ordered with respect to the $\preceq_x$ relationship and succinct, by comparing the pair to be inserted with the last pair only. S is computed by performing the union between S_l and S_r as described in Lemma 5.4. The algorithm iterates until all the children

Algorithm 6 TREE_PARALLEL

Require: A P-node μ, thickness $t(\nu_i)$ and pairs $\langle x_i, 0\rangle, \langle 0, y_i\rangle \in \hat{A}(\nu_i)$ of each child component ν_i, for $i = 1, \ldots, \delta(\mu)$, and the gist $\hat{A}(\nu_c)$ of the admissible set of the child ν_c chosen as the center of the permutation.

Ensure: The gist $\hat{A}_{\nu_c}(\mu)$ of the admissible set restricted to all the permutations having ν_c as center.

1: $S = \hat{A}(\nu_c)$;
2: **for** $i = 1$ to $\delta(\mu)$ **do**
3: $p_l = p_r = \langle \infty, \infty \rangle$;
4: **for all** $p^k = \langle x_k, y_k \rangle \in S$ **do**
5: **if** $p_l \not\preceq \langle \max(x_k + t(\nu_i), x_i), y_k \rangle$ **then**
6: $p_l = \langle \max(x_k + t(\nu_i), x_i), y_k \rangle$;
7: S_l.addLast(p_l);
8: **end if**
9: **if** $p_r \not\preceq \langle x_k, \max(y_k + t(\nu_i), y_i) \rangle$ **then**
10: $p_r = \langle x_k, \max(y_k + t(\nu_i), y_i) \rangle$;
11: S_r.addLast(p_r);
12: **end if**
13: **end for**
14: $S = S_l \cup S_r$;
15: **end for**
16: **return** S;

components have been added.

Lemma 5.22 *Starting from $\hat{A}(\nu_i)$ and $t(\nu_i)$, Algorithm* TREE_PARALLEL *computes the gist $\hat{A}_{\nu_c}(\mu)$ of the admissible set of μ, restricted to all the permutations having ν_c as center, in $O(\sum_{k=1}^{\delta(\mu)} \sum_{i=1}^{k} n(\nu_i))$ time.*

Proof: When the k-th component is added, the computational complexity of the construction of S_l and S_r and of their union depends on the number of nodes of $T(\nu_c)$ at distance k from the root, which is, by definition, $O(\sum_{i=1}^{k} n(\nu_i))$. Since such operations have to be performed for each level, the overall time complexity bound follows. □

Lemma 5.23 *Starting from $\hat{A}(\nu_i)$ and $t(\nu_i)$, for $i = 1, \ldots, \delta(\mu)$, the gist $\hat{A}(\mu)$ of the admissible set of μ can be computed in time $O(\delta(\mu) \cdot \sum_{k=1}^{\delta(\mu)} \sum_{i=1}^{k} n(\nu_i))$.*

Proof: In order to compute $\hat{A}(\mu)$ we have to perform a sequence of computations. The proof is based on the fact that the heaviest of these computations needs $O(\delta(\mu) \cdot \sum_{k=1}^{\delta(\mu)} \sum_{i=1}^{k} (n(\nu_i)))$ time. By Lemma 5.18, the thickness $t(\mu)$ can be computed in $O(\delta(\mu))$ time. The gists $\hat{A}_{\nu_c}(\mu)$, for $c = 1, \ldots, \delta(\mu)$, of the admissible sets restricted to all the permutations having ν_c as center can be computed, by Lemma 5.22, in $O(\delta(\mu) \cdot \sum_{k=1}^{\delta(\mu)} \sum_{i=1}^{k} (n(\nu_i)))$ total time. Since, by Property 5.5, for each gist $\hat{A}_{\nu_c}(\mu)$ we have $|\hat{A}_{\nu_c}(\mu)| = O(n)$, the gist $\hat{A}(\mu) = \cup_{c=1}^{\delta(\mu)} \hat{A}_{\nu_c}(\mu)$ of the admissible set of μ can be computed with Algorithm GIST_UNION in $O(\delta(\mu) \cdot n)$ time, according to Lemma 5.4. □

Computing the Minimum Depth and the Minimum-Depth Embedding

At the end of the bottom-up traversal of $\mathcal{T}$, the value of the minimum depth can be computed starting from the gist of the admissible set of the component μ that is the child of the root e of $\mathcal{T}$. Namely, for each pair $\langle x_h, y_h \rangle \in \hat{A}(\mu)$, let m_h be $x_h + 1$ if $x_h = y_h$, and let m_h be $\max(x_h, y_h)$ otherwise. The minimum depth is the minimum of the m_h, for $h = 1, \ldots, |\hat{A}(\mu)|$.

Lemma 5.24 *Let $\mathcal{T}$ be the SPQR-tree rooted at edge e of an n-vertex graph G, and let μ be the child of e. Starting from $\hat{A}(\mu)$, the minimum depth of the embeddings of G with e on the external face can be computed in $O(n)$ time.*

Proof: Consider a pair $\langle x_h, y_h \rangle \in \hat{A}(\mu)$. By definition, there exists an embedding Γ_μ^h such that each face $f \in \Gamma_\mu^h$ is at distance less or equal than x_h from f_l^μ or at distance less or equal than y_h from f_r^μ. Two planar embeddings of G with e on the external face are obtained from Γ_μ^h by adding edge e between the poles of μ, and by choosing either f_l^μ or f_r^μ as the external face f^μ, respectively. If $x_h > y_h$ we set $f^\mu = f_l^\mu$ and $m_h = x_h$. If $x_h = y_h$ we set $f^\mu = f_l^\mu$ and $m_h = x_h + 1$. Otherwise, if $x_h < y_h$ we set $f^\mu = f_r^\mu$ and $m_h = y_h$. Denote Γ_G^h the corresponding embedding.

We have that if $x_h > y_h$, each face $f \in \Gamma_G^h$ is at distance from f^μ less or equal than $\max(x_h, y_h + 1) = x_h = m_h$. If $x_h = y_h$ each face $f \in \Gamma_G^h$ is at distance from f^μ less or equal than $\max(x_h, y_h + 1) = x_h + 1 = m_h$. If $x_h < y_h$ each face $f \in \Gamma_G^h$ is at distance from f^μ less or equal than $\max(x_h + 1, y_h) = y_h = m_h$.

Since $\hat{A}(\mu)$ accounts for the depths of all possible embeddings of the pertinent graph G_μ, the minimum among all the m_h is the minimum depth of the embeddings of G with e on the external face. The operations to be performed are $|\hat{A}(\mu)|$ integer comparisons and the computation of the minimum among $|\hat{A}(\mu)|$ integer values. Since, by Property 5.5, $|\hat{A}(\mu)|$ is $O(n)$, the statement follows. □

Theorem 5.1 *Let G be an n-vertex biconnected planar graph and let $\mathcal{T}$ be the SPQR-tree of G rooted at e. The minimum depth of an embedding of G with e on the external face can be computed in $O(n^3)$ time and $O(n^2)$ space.*

Proof: Consider the three sets S, R and P containing the series, rigid, and parallel nodes of $\mathcal{T}$, respectively. For each series component $\mu_s \in S$, by Lemmas 5.11 and 5.13, $t(\mu_s)$ and $\hat{A}(\mu_s)$ can be computed in $O(\sum_{i=1}^{\delta(\mu_s)} n(\nu_i))$ time. Hence, the overall complexity for all the series nodes is $O(\sum_{\mu_s \in S} \sum_{i=1}^{\delta(\mu_s)} n(\nu_i))$. Since the number of the series nodes is $O(n)$, the above sum is $O(n^2)$.

For each rigid component $\mu_r \in R$, by Lemmas 5.14 and 5.17, $t(\mu_r)$ and $\hat{A}(\mu_r)$ can be computed in $O(\sum_{j=1}^{\delta(\mu_r)} \sum_{i=1}^{j} n(\nu_i))$. Hence, the overall complexity for all the rigid nodes is $O(\sum_{\mu_r \in R} \overbrace{\sum_{j=1}^{\delta(\mu_r)}}^{O(n)} \overbrace{\sum_{i=1}^{j} n(\nu_i)}^{O(n)})$, which is $O(n^2)$, since the total number of children of all the rigid nodes is less or equal than the total number of arcs of $\mathcal{T}$, that is $O(n)$.

For each parallel component $\mu_p \in P$, by Lemmas 5.18 and 5.23, $t(\mu_p)$ and $\hat{A}(\mu_p)$ can be computed in $O(\delta(\mu_p) \sum_{k=1}^{\delta(\mu_p)} \sum_{i=1}^{k} n(\nu_i))$ time. Hence, the overall complexity for all the parallel nodes is $O(\sum_{\mu_p \in P} \overbrace{\delta(\mu_p) \sum_{k=1}^{\delta(\mu_p)}}^{O(n^2)} \overbrace{\sum_{i=1}^{k} n(\nu_i)}^{O(n)})$, which is $O(n^3)$, since the total number of children of all the parallel nodes is $O(n)$. The time complexity of the bottom-up traversal is $O(n^2)+O(n^2)+O(n^3) = O(n^3)$. Starting from the gist of the admissible set of the root, the minimum depth is computed, by Lemma 5.24, in $O(n)$ time.

The space bound can be obtained by considering that there are $O(n)$ components in $\mathcal{T}$ and that, by Property 5.5, the size of the gists of their admissible sets is $O(n)$. □

To produce a minimum-depth embedding of G with an edge e on the external face we need some additional information to be added to each component during the bottom-up traversal of $\mathcal{T}$, meant to describe how the components must be attached together in order to obtain an embedding satisfying each pair of the gist of the admissible set.

Namely, for each node μ and for each pair $p \in \hat{A}(\mu)$ we attach an "embedding descriptor" composed of :

- A Boolean variable b_μ specifying how μ must be attached to its parent component ν. Namely, consider the two faces f_l and f_r of $sk(\nu)$ incident to the virtual edge e_μ representing μ in $sk(\nu)$. Variable b_μ describes whether μ must replace e_μ in $sk(\nu)$ with f_l^μ corresponding to f_l or not.
- An integer pair p_i for each child component ν_i of μ specifying how ν_i must be embedded to obtain an embedding of μ satisfying p.
- If μ is a parallel component we also record the needed ordering of its child components ν_i.

The minimum-depth embedding is computed with a top-down traversal of the SPQR-tree $\mathcal{T}$ rooted at e, using the above described additional structures, by suitably replacing each virtual edge with the skeleton of the corresponding component.

Theorem 5.2 *Let G be an n-vertex biconnected planar graph. A minimum-depth embedding of G can be computed in $O(n^4)$ time and $O(n^3)$ space.*

Proof: For each edge e of G, compute the SPQR-tree rooted at e in $O(n)$ time and the minimum-depth embedding with e on the external face in $O(n^3)$ time. The cubic space bound is due to the fact that, for each component and for each integer pair of the gist of its admissible set, an integer pair for each children must be recorded. □

5.6 Extension to General Planar Graphs

The minimum-depth embedding of a simply-connected planar graph G, described by its BC-tree, can be found with an approach similar to that used in [BM90]. The key point of such an approach is that the algorithm to compute a minimum-depth embedding of a biconnected graph with a specified edge on the external face can be suitably modified in order to be applied to each

block ρ_i, taking into account the depth of the blocks that are attached to the vertices of ρ_i that are cut-vertices of G, while maintaining the $O(n_i^3)$ time complexity, where n_i is the number of vertices of ρ_i.

In order to compute the minimum-depth embedding of G, we perform a separate computation with each edge e on the external face, rooting the BC-tree at the block containing e. We traverse bottom-up the BC-tree applying the algorithm sketched in the following. Each block ρ_j, sharing cut-vertex v_j with its parent ρ_i, must be embedded with v_j on its external face. Hence, we apply the modified algorithm to ρ_j using as reference edge each one of the edges incident to v_j and choose the embedding with minimum depth.

This computation has to be performed for each edge of each block chosen as the root block. The overall $O(n^4)$ complexity can be obtained by considering that the modified algorithm has to be launched at most three times for each edge of G. Namely, we launch the algorithm on each edge e of G when such an edge is chosen to be on the external face, taking into account the depths of all the attached blocks, and we launch the algorithm on each edge e incident to a cut-vertex v (hence, at most two times for each e) taking into account the depths of all the attached blocks with the exception of those attached to v.

Theorem 5.3 *Let G be an n-vertex connected planar graph. A minimum-depth embedding of G can be computed in $O(n^4)$ time and $O(n^3)$ space.*

Here we provide a sketch of the algorithm to find a minimum-depth embedding of a block ρ of a BC-tree $\mathcal{B}$ with a specified edge e on the external face that takes into account the depths of the child blocks π_i sharing a cut-vertex v_i with ρ. The input of such an algorithm is a minimum-depth embedding for each child block π_i of ρ with v_i on the external face. The algorithm described in Section 5.5 for computing a minimum-depth embedding of a biconnected graph with a specified edge on the external face has to be modified in order to take into account the fact that each child block π_i has to be placed inside one of the faces of ρ incident to v_i. Consider the skeleton of a series, rigid, or parallel component μ of the SPQR-tree of the block ρ. The child blocks of ρ, whose depths have to be taken into account, may only attach to the poles of the child components of μ that are not poles of μ. In fact, child blocks attached to the poles of μ will be taken into account when the parent of μ is considered by the algorithm. Hence, the computation for a parallel component μ_p is unchanged with respect of the algorithm described in Section 5.5.

The key observation for modifying the algorithm for a series or a rigid component μ is that the admissible set of $sk(\mu)$ and the admissible sets of each

child component ν_i nested into $\Gamma^j(\mu)$ do not change whichever will be the faces of $sk(\mu)$ that will be chosen to contain the child blocks π_i attached to ρ.

The second key observation is that each block π_i, with depth d_i, can be modeled by means of an additional admissible set representing the contribution of the block on the depth of the whole component. Namely, for a series component μ_s, it is sufficient to intersect the gists of the admissible sets $\hat{A}(\nu_h|\Gamma^j(\mu))$ computed for each child component ν_h of μ with the additional admissible set containing the pairs $\langle 0, d_i\rangle$ and $\langle d_i, 0\rangle$, which represent the fact that the block π_i will be placed into $f_r^{\mu_s}$ or $f_l^{\mu_s}$, respectively. Analogously, the algorithm for a rigid component μ_r must be modified by considering for each child block π_i, with depth d_i, an additional admissible set representing the fact that π_i can be placed in any face incident to v_i. Such a set contains two pairs $\langle 0, d_i + d_r(f)\rangle$ and $\langle d_i + d_l(f), 0\rangle$ for each face incident to v_i (its gist will contain two pairs only). It can be proved that the above changes do not affect the overall $O(n^3)$ complexity of the algorithm, where n is the number of vertices of ρ.

5.7 Conclusions

In this chapter we presented an $O(n^4)$-time algorithm for computing a minimum-depth embedding of a planar graph.

Since our approach is inspired by the one in [BM90], it is useful to stress the similarities and the differences between the two contributions. We take from [BM90] the fundamental idea of decomposing the graph into components and to separately consider each component. Also, the concept of *thickness* is the same as in [BM90]. Both the approaches exploit the strategy of equipping each component with pairs of integers, representing their distance from the external face. However, in [BM90] a pair represents the result of a "probe" that says that a certain component is feasible with that depth. In our case a set of pairs represents implicitly all the admissible values of depth of the component. The combinatorial structure of such pairs and their nice computational properties are a key ingredient of our approach. The techniques for combining the components are similar. However, in the critical problem of dealing with parallel compositions, we develop an approach that has many new features.

The natural problem that remains open is to fill the gap from our $O(n^4)$ time to the linear time obtained in [PT00] for a simplified version of the problem. Also, it would be interesting to understand whether the techniques used in this chapter could be applied to improve the complexity bounds for the other distance measures, such as radius, outerplanarity, and width.

Part III

Greedy Drawings of Planar Graphs

Chapter 6

Greedy Drawings of Planar Graphs

In this chapter[1], we study greedy drawings of planar graphs, a type of drawings that have to satisfy some geometrical properties related to the distance among the vertices. Namely, in such a type of drawings, every vertex is connected to any other vertex by a path such that at each step the geometric distance to the destination is decreased. This problem has recently been the subject of some investigation, as it is motivated by its application in greedy routing algorithms for sensor networks.

We study the problem aiming at a combinatorial characterization of the graphs admitting a drawing with this property. In this context, we first present an algorithm to construct greedy drawings of every given triangulation and then we show how to extend it to work for every given triconnected planar graph, hence proving a conjecture by Papadimitriou and Ratajczak [PR05], that was independently proved by Leighton and Moitra [LM08].

Our algorithm for triangulations relies on two main results. First, we show how to construct greedy drawings of a fairly simple class of graphs, called *triangulated binary cactuses*. Second, we show that every triangulation can be spanned by a triangulated binary cactus. The technique used for triconnected planar graphs is the same, but in this case we use a slightly different class of graphs, called *non-triangulated binary cactuses*.

[1] Part of the contents of this chapter are a joint work with Fabrizio Frati and Luca Grilli, appeared in [AFG08] and to appear in [AFG10]. Thanks to Tom Leighton and Ankur Moitra for providing us with their paper, for introducing us to Gao and Richter's results, and for helping us to clarify the relationship between our results, their own, and Gao and Richter's ones.

6.1 Introduction

The standard Internet routing protocol is as follows: Each computer is univocally identified by an *IP-address*; IP-addresses are *aggregated*, i.e., computers that are topologically or geographically close in the network are assigned addresses with the same most significative bits; consequently, routers do not have to know the route to each address in the network, but they maintain in their *routing tables* only the information on the route to take for reaching each set of aggregated addresses. Such an approach does not work in many wireless networks, such as *ad-hoc* and *sensor networks*, where the addresses that are assigned to nodes geographically or topologically close are not necessarily similar. Despite of their importance, no universally-accepted communication protocol exists for such wireless environments.

Geographic routing is a class of routing protocols in which nodes forward packets based on their geographic locations. Among such protocols, *geometric routing*, or *greedy routing*, has been well investigated, because it relies on a very simple strategy in which, in order to forward packets, each node has to know only local information and the destination address. In fact, in the greedy routing a node forwards packets to a neighbor that is *closer than itself* to the destination's geographic location. Different distance metrics define different meanings for the word "closer", and consequently define different routing algorithms for the packet delivery. The most used and studied metric is of course the *Euclidean distance*. The efficiency of geographic routing algorithms strongly relies on the geographic coordinates of the nodes. This is indeed a drawback of such routing algorithms, for the following reasons: (i) Nodes of the network have to know their locations, hence they have to be equipped with GPS devices, which are expensive and increase the energy consumption of the nodes; (ii) geographic coordinates are independent of the network obstructions, i.e. obstacles making the communication between two close nodes impossible, and, more in general, they are independent of the network topology; this could lead to situations in which the communication fails because a *void* has been reached, i.e., the packet has reached a node whose neighbors are all farther from the destination than the node itself.

A brilliant solution to the geographic routing weakness has been proposed by Rao, Papadimitriou, Shenker and Stoica, who in [RPSS03] proposed a scheme in which nodes decide *virtual coordinates* and then apply the standard geometric routing algorithm relying on such virtual locations rather than on the real geographic coordinates. Clearly, virtual coordinates do not need to reflect the nodes actual positions, hence they can be suitably chosen to guaran-

tee that the geometric routing algorithm delivers packets with high probability. It has been experimentally shown that such an approach strongly improves the reliability of geometric routing [RPSS03, PR05].

Unfortunately, greedy routing in the Euclidean plane is not always possible, as for example if the network is represented as a star graph with seven vertices. Thus, in order to apply greedy routing to arbitrary networks, researchers have had to either abandon the natural geometry of the Euclidean plane and use hyperbolic coordinates or to abandon the simple greedy routing protocol. Namely, it has been proved that virtual coordinates guarantee geometric routing to work for every connected topology when they can be chosen in the hyperbolic plane [Kle07], even if only a logarithmic number of bits are available to store the coordinates of each node [EG08]. Moreover, some easy modifications of the routing algorithm guarantee that Euclidean virtual coordinates can be chosen so that the packet delivery always succeeds [BCGG06], even if the coordinates need to be locally computed [BCGW07].

Concerning greedy routing in the Euclidean plane, the most intense research effort has been devoted to determining network topologies for which greedy routing with Euclidean virtual coordinates is guaranteed to work. From a graph-theoretic point of view, the problem can be restated as follows: Which graphs admit a *greedy drawing*, i.e., a straight-line drawing Γ in the plane such that, for every pair of nodes, there exists a *distance-decreasing path* in Γ? A path $(v_0, v_1, \ldots, v_m)$ is distance-decreasing if $d(v_i, v_m) < d(v_{i-1}, v_m)$, for $i = 1, \ldots, m$, where $d(u, v)$ denotes the Euclidean distance between points u and v. This formulation of the problem gives a clear perception of how greedy routing can be seen as a "bridge" problem between the theory of routing and Graph Drawing, thus explaining why it attracted attention in both areas.

In [PR05] Papadimitriou and Ratajczak conjectured the following:

Conjecture 6.1 *Every triconnected planar graph admits a greedy drawing.*

Papadimitriou and Ratajczak observed that, if a graph G admits a greedy drawing, then any graph containing G as a spanning subgraph admits one, as well. It follows that Conjecture 6.1 extends to all graphs which are spanned by a triconnected planar graph. Related to such an observation, Papadimitriou and Ratajczak proved that every triconnected graph not containing any $K_{3,3}$-minor has a triconnected planar spanning subgraph.

Some examples of graphs not admitting any greedy drawings in the plane adopting the Euclidean distance have been provided. Namely, Papadimitriou and Ratajczak [PR05] proved that $K_{k,5k+1}$ has no greedy drawing, for $k \geq 1$.

This result is interesting when related to Conjecture 6.1, as it states that $K_{2,11}$, that is planar but non-triconnected, and $K_{3,16}$, that is triconnected but non-planar, do not admit any greedy drawing, hence making the two conditions of the conjecture necessary. Also, in [LM08] it is proved that there exist trees not admitting any greedy drawing, as the complete binary tree with 31 vertices.

There are a few classes of triconnected planar graphs for which the conjecture is easily shown to be true, for example graphs with a *Hamiltonian path* and *Delaunay Triangulations*. At SODA 2008 [Dha08], Dhandapani proved the conjecture for the first non-trivial class of triconnected planar graphs, namely he showed that every *triangulation* admits a greedy embedding. The proof of Dhandapani is probabilistic, namely the author proves that, for every given triangulation G, a *random Schnyder drawing* of G [Sch90] is greedy with positive probability; hence, there exists a greedy drawing of every triangulation. Although such a proof is elegant, relying at the same time on an old Combinatorial Geometry theorem, known as the *Knaster-Kuratowski-Mazurkievicz Theorem* [KKM29], and on standard Graph Drawing techniques, as the *Schnyder realizers* [Sch90] and the *canonical orderings* of a triangulation [dPP90], it does not lead to an embedding algorithm.

In this chapter we show an algorithm for constructing greedy drawings of triangulations. We define a simple class of graphs, called *triangulated binary cactuses*, and we provide an algorithm to construct a greedy drawing of any such a graph. Further, we show how to find, for every triangulation, a triangulated binary cactus spanning it. It is clear that the previous statements imply an algorithm for constructing greedy drawings of triangulations. Namely, consider any triangulation G, apply the algorithm to find a triangulated binary cactus S spanning G, and then apply the algorithm to construct a greedy drawing of S. As already observed, adding edges to a greedy drawing leaves the drawing greedy, hence S can be augmented to G, obtaining a greedy drawing of G.

Theorem 6.1 *Given a triangulation G, there exists an algorithm to compute a greedy drawing of G.*

Further, we provide an algorithm to construct greedy drawings of general triconnected planar graphs. The strategy of such an algorithm is the same as the one of the algorithm for constructing greedy drawings of triangulations. In fact, we define a simple class of graphs, called *non-triangulated binary cactuses*, and we provide an algorithm to construct a greedy drawing of any such a graph. Finally, we show how to find, for every triconnected planar graph, a non-triangulated binary cactus spanning it. Such a result proves Conjecture 6.1;

however, the conjecture has been very recently (and independently) proved by Leighton and Moitra [LM08], by using techniques which are amazingly similar to ours. Hence, we will only sketch how to modify the algorithm we provide for triangulations in order to make it work for general triconnected planar graphs and we will extensively discuss differences and similarities of our algorithm with respect to Leighton and Moitra's.

The rest of the chapter is organized as follows. In Sect. 6.2 we introduce triangulated binary cactuses; in Sect. 6.3 we show an algorithm to construct greedy drawings of triangulated binary cactuses; in Sect. 6.4 we show an algorithm to construct a triangulated binary cactus spanning a given triangulation; in Sect. 6.5 we show how to modify the algorithm described for triangulations in order to make it work for general triconnected planar graphs and we compare our techniques with Leighton and Moitra's ones; finally, in Sect. 6.6 we conclude and present some open problems concerning greedy graph drawings.

6.2 Triangulated Binary Cactuses

Consider a graph G. Consider its BC-tree $\mathcal{T}$ and suppose it is rooted at a specific B-node corresponding to a block ν. When the BC-tree $\mathcal{T}$ of a graph G is rooted at a certain block ν, we denote by $G(\mu)$ the subgraph of G induced by all the vertices in the blocks contained in the subtree of $\mathcal{T}$ rooted at μ. In a rooted BC-tree $\mathcal{T}$ of a graph G, for each B-node μ we denote by $r(\mu)$ the cutvertex of G parent of μ in $\mathcal{T}$. If μ is the root of $\mathcal{T}$, i.e., $\mu = \nu$, then we let $r(\mu)$ denote any non-cutvertex node of the block associated with μ. In the following, unless otherwise specified, each considered BC-tree is meant to be rooted at a certain B-node ν such that the block associated with ν has at least one vertex $r(\nu)$ which is not a cutvertex. It is not difficult to see that such a block exists in every planar graph.

A *triangulated binary cactus* S, in the following two sections simply called *binary cactus*, is a connected graph such that (see Fig 6.1):

- The block associated with each B-node of $\mathcal{T}$ is either an edge or a *triangulated cycle*, i.e., a cycle $(r(\mu), u_1, u_2, \ldots, u_h)$ triangulated by the edges from $r(\mu)$ to each of $u_1, u_2, \ldots, u_h$.

- Every cutvertex is shared by exactly two blocks of S.

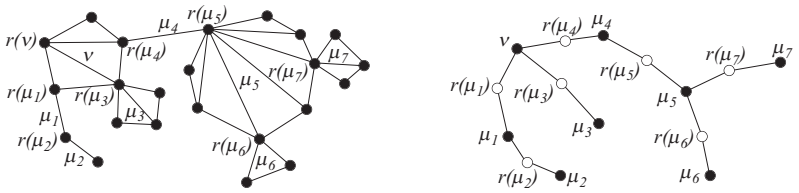


Figure 6.1: (a) A binary cactus S. (b) The block-cutvertex tree of S. White (resp. black) circles represent C-nodes (resp. B-nodes).

6.3 Greedy Drawings of Binary Cactuses

In this section, we give an algorithm to compute a greedy drawing of a binary cactus S. Such a drawing is constructed by performing a bottom-up traversal of the BC-tree $\mathcal{T}$ of S.

Consider the root μ of a subtree of $\mathcal{T}$ corresponding to a block of S, consider the k children of μ, which correspond to cutvertices of S, and consider the children of such cutvertices, say $\mu_1, \mu_2, \ldots, \mu_k$. Notice that each C-node child of μ is parent of exactly one B-node μ_i of $\mathcal{T}$, by the definition of binary cactus. For each $i = 1, \ldots, k$, inductively assume there is a drawing Γ_i of $S(\mu_i)$ satisfying the properties listed below.

Let C be a circle, let (a_i, b_i) be an arc of C, let p_i^* be a point of C such that the diameter through p_i^* cuts (a_i, b_i) in two arcs of the same length. Let α_i and β_i be any two angles such that $\alpha_i \leq \beta_i \leq \frac{\pi}{4}$. Consider the tangent $t(p_i^*)$ to C in p_i^*. Consider two half-lines l_1^* and l_2^* incident to p_i^*, lying on the opposite part of C with respect to $t(p_i^*)$, and forming angles equal to β_i with $t(p_i^*)$. Denote by $W(p_i^*)$ the wedge centered at p_i^*, delimited by l_1^* and l_2^*, and not containing C. Refer to Fig. 6.2(a).

- *Property 1.* Γ_i is a greedy drawing.
- *Property 2.* Γ_i is entirely contained inside a region $R(\Gamma_i)$ delimited by arc (a_i, b_i), and by segments $\overline{p_i^* a_i}$ and $\overline{p_i^* b_i}$. The angle $\widehat{a_i p_i^* b_i}$ is α_i.
- *Property 3.* For every vertex v in $S(\mu_i)$ and for every point p internal to $W(p_i^*)$, there exists in Γ_i a path $(v = v_0, v_1, \ldots, v_l = r(\mu_i))$ from v to $r(\mu_i)$ such that $d(v_j, p) < d(v_{j-1}, p)$, for $j = 1, \ldots, l$.

- *Property 4.* For every vertex v in $S(\mu_i)$ and for every point p internal to $W(p_i^*)$, $d(v, p_i^*) < d(v, p)$.

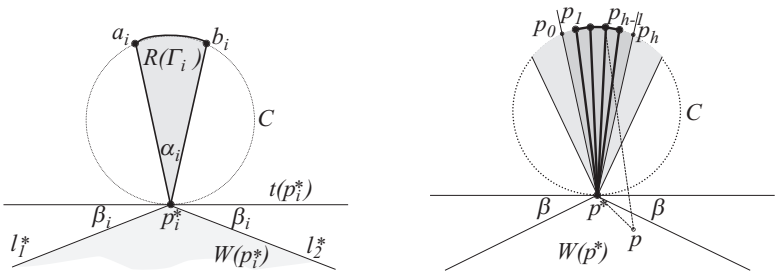


Figure 6.2: (a) Illustration for Properties 1–4 of Γ_i. (b) Base case of the algorithm. The light and dark shaded region represents $R(\Gamma)$ (the angle of $R(\Gamma)$ at p^* is α). The dark shaded region represents the intersection of $W(p^*, \frac{\alpha}{2})$ with the disk delimited by C.

In the base case, block μ has no child. Denote by $(r(\mu) = u_0, u_1, \ldots, u_{h-1})$ the block of S corresponding to μ. Notice that $h \geq 2$. Consider any circle C with center c. Let p^* be the point of C with smallest y-coordinate. Consider the wedges $W(p^*, \alpha)$ and $W(p^*, \frac{\alpha}{2})$ with angles α and $\frac{\alpha}{2}$, respectively, incident to p^* and such that the diameter of C through p^* is their bisector (see Fig. 6.2(b)). Place $r(\mu)$ at p^*. Denote by p'_a and p'_b the intersection points (different from p^*) of the half-lines delimiting $W(p^*, \frac{\alpha}{2})$ with C. Denote by A the arc of C between p'_a and p'_b and not containing p^*. Consider $h+1$ points $p_0, p_1, \ldots, p_h$ on A such that $p_0 = p'_a$, $p_h = p'_b$, and the distance between any two consecutive points p_i and p_{i+1} is the same. Place vertex u_i at point p_i, for $i = 1, 2, \ldots, h-1$. Notice that, if $h = 2$, μ corresponds to an edge of S that is drawn as a vertical segment, with u_1 above u_0.

In order to show that the constructed drawing Γ satisfies Property 1, consider any two vertices u_i and u_j, with $i < j$. If $i = 0$, then u_0 and u_j are joined by an edge, which provides a distance-decreasing path between them. Otherwise, we prove that $(u_i, u_{i+1}, \ldots, u_j)$ is a distance-decreasing path from u_i to u_j, the proof that $(u_j, u_{j-1}, \ldots, u_i)$ is a distance-decreasing path from u_j to u_i being analogous. For each $l = i, i+1, \ldots, j-2$, angle $u_l\widehat{u_{l+1}}u_j$ is greater than $\frac{\pi}{2}$, because triangle (u_l, u_{l+1}, u_j) is inscribed in less than half a circle with u_{l+1} as middle point (see Fig. 6.3(a)). Hence, (u_l, u_j) is the longest

side of triangle (u_l, u_{l+1}, u_j) and $d(u_{l+1}, u_j) < d(u_l, u_j)$ follows. Drawing Γ satisfies Property 2 by construction. In order to prove that Γ satisfies Property 3, we have to prove that, for every vertex u_i, with $i \geq 1$, and for every point p in $W(p^*)$, $d(u_0, p) < d(u_i, p)$. Angle $\widehat{pp^*p_i}$ is at least $\beta + (\frac{\pi}{2} - \frac{\alpha}{4})$, which is greater than $\frac{\pi}{2}$ (see Fig. 6.3(b)). It follows that segment $\overline{pp_i}$ is the longest side of triangle (p, p^*, p_i), thus proving that $d(u_0, p) < d(u_i, p)$. For the same reason $d(u_0, u_i) < d(p, u_i)$, hence proving Property 4.

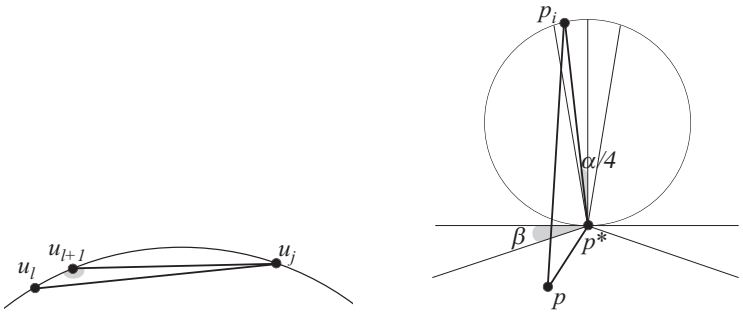


Figure 6.3: (a) Γ satisfies Property 1. (b) Γ satisfies Properties 3 and 4.

Now we discuss the inductive case. Suppose that μ is a node of $\mathcal{T}$ having k children. We show how to construct a drawing Γ of $S(\mu)$ satisfying Properties 1–4 with parameters α and β. Refer to Fig. 6.4. Denote by $(r(\mu) = u_0, u_1, \ldots, u_{h-1})$ the block of S corresponding to μ. Remember that $h \geq 2$ and that if $h = 2$, then the block is an edge, otherwise it is a triangulated cycle. Consider any circle C with center c. Let p^* be the point of C with smallest y-coordinate. Consider the wedges $W(p^*, \alpha)$ and $W(p^*, \frac{\alpha}{2})$ with angles α and $\frac{\alpha}{2}$, respectively, incident to p^* and such that the diameter of C through p^* is their bisector. Region $R(\Gamma)$ is the intersection region of $W(p^*, \alpha)$ with the closed disk delimited by C.

Consider a second circle C' with center c intersecting the two lines delimiting $W(p^*, \frac{\alpha}{2})$ in two points p'_a and p'_b such that angle $\widehat{p'_a c p'_b} = \frac{3\alpha}{2}$. It is not difficult to see that such a circle always exists. Denote by A the arc of C' delimited by p'_a and p'_b and farther from p^*. Consider $h + 1$ points $p_0, p_1, \ldots, p_h$ on A such that $p_0 = p'_a$ and $p_h = p'_b$, and the distance between any two consecutive points p_i and p_{i+1} is the same. Observe that, for each $i = 0, 1, \ldots, h-1$, angle $\widehat{p_i c p_{i+1}} = \frac{3\alpha}{2h}$.

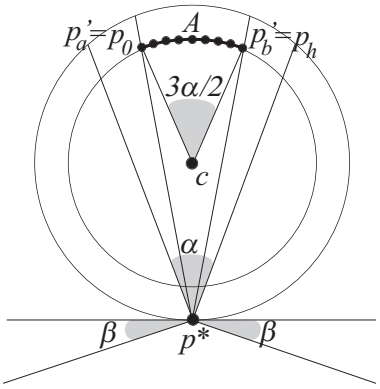


Figure 6.4: Construction of a drawing Γ in the inductive case of the algorithm.

First, we draw the block of S corresponding to μ. As in the base case, place vertex $u_0 = r(\mu)$ at p^* and, for $i = 1, 2, \ldots, h-1$, place u_i at point p_i. Recursively construct a drawing Γ_i of $S(\mu_i)$ satisfying Properties 1–4 with $\alpha_i = \frac{3\alpha}{16h}$ and $\beta_i = \frac{3\alpha}{8h}$.

We are going to place each drawing Γ_i of $S(\mu_i)$ together with the constructed drawing of the block of S corresponding to μ, thus obtaining a drawing Γ of $S(\mu)$. Notice that not all the h nodes u_i are cutvertices of S. However, with a slight abuse of notation, we suppose that block $S(\mu_i)$ has to be placed at node u_i. Refer to Fig 6.5. Consider point p_i and its "neighbors" p_{i-1} and p_{i+1}, for $i = 1, 2, \ldots, h-1$. Consider lines $t(p_{i-1})$ and $t(p_{i+1})$ tangent to C' in p_{i-1} and p_{i+1}, respectively. Further, consider circles C_{i-1} and C_{i+1} centered at p_{i-1} and p_{i+1}, respectively, and passing through p_i. Moreover, consider lines h_{i-1} and h_{i+1} tangent to C_{i-1} and C_{i+1} in p_i, respectively. For each point p_i, with $i = 0, \ldots, h$, consider two half-lines t_1^i and t_2^i incident to p_i, forming angles $\beta_i = \frac{3\alpha}{8h}$ with $t(p_i)$, and both lying in the half-plane delimited by $t(p_i)$ and containing C'. Denote by $W(p_i)$ the wedge delimited by t_1^i and by t_2^i, and containing c.

We will place Γ_i inside (a part of) the bounded region R_i obtained as the intersection of: (i) the half-plane H^{i-1} delimited by h_{i-1} and not containing C_{i-1}, (ii) the half-plane H^{i+1} delimited by h_{i+1} and not containing C_{i+1}, (iii) wedge $W(p_{i-1})$, (iv) wedge $W(p_{i+1})$, and (v) the disk delimited by C.

First, we prove that R_i is "large enough" to contain Γ_i, namely we claim that there exists an isosceles triangle T that has an angle larger than $\alpha_i = \frac{3\alpha}{16h}$

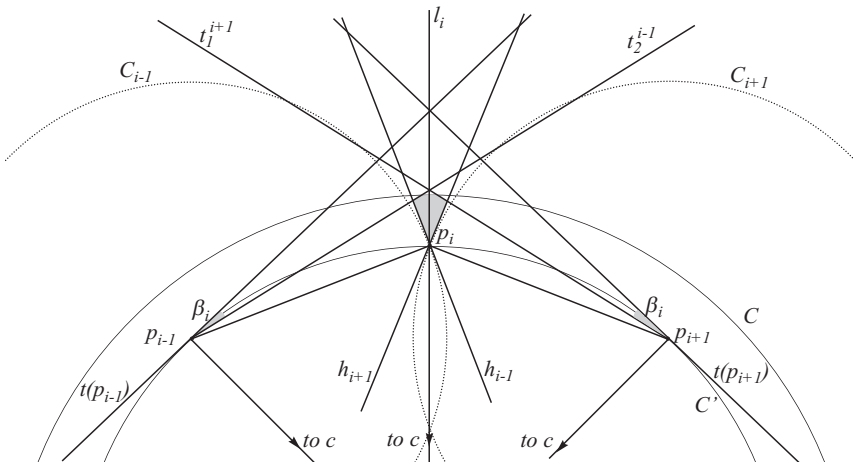


Figure 6.5: Lines and circles in the construction of Γ. The shaded areas represent angles β_i and region R_i.

incident to p_i and that is completely contained in R_i. Such a triangle will have the further feature that the angle incident to p_i is bisected by the line l_i through c and p_i.

Lines h_{i-1} and h_{i+1} are both passing through p_i; we prove that they have different slopes and we compute the angles that they form at p_i. Refer to Fig. 6.6. Line h_{i-1} forms an angle of $\frac{\pi}{2}$ with segment $\overline{p_{i-1}p_i}$; angle $\widehat{cp_ip_{i-1}}$ is equal to $\frac{\pi}{2} - \frac{3\alpha}{4h}$, since $\widehat{p_icp_{i-1}} = \frac{3\alpha}{2h}$ and since triangle (p_{i-1}, c, p_i) is isosceles. Hence, the angle delimited by h_{i-1} and l_i is $\pi - \frac{\pi}{2} - (\frac{\pi}{2} - \frac{3\alpha}{4h}) = \frac{3\alpha}{4h}$. Analogously, the angle between l_i and h_{i+1} is $\frac{3\alpha}{4h}$. Hence, the intersection of H^{i-1} and H^{i+1} is a wedge $W(p_i, h_{i-1}, h_{i+1})$ centered at p_i, with an angle of $\frac{3\alpha}{2h}$, and bisected by l_i.

We claim that each of t_2^{i-1} and t_1^{i+1} cuts the border of $W(p_i, h_{i-1}, h_{i+1})$ twice. The angle between $t(p_{i-1})$ and $\overline{p_{i-1}p_i}$ is $\frac{3\alpha}{4h}$, because the angle between $t(p_{i-1})$ and $\overline{cp_{i-1}}$ is $\frac{\pi}{2}$, and angle $\widehat{cp_{i-1}p_i}$ is $\frac{\pi}{2} - \frac{3\alpha}{4h}$. The angle between $t(p_{i-1})$ and t_2^{i-1} is $\beta_i = \frac{3\alpha}{8h}$, by construction. Hence, the angle between t_2^{i-1} and $\overline{p_{i-1}p_i}$ is $\frac{3\alpha}{4h} - \frac{3\alpha}{8h} = \frac{3\alpha}{8h}$. Since the slope of both h_{i-1} and h_{i+1} with respect to $\overline{p_{i-1}p_i}$ is greater than $\frac{3\alpha}{8h}$ and smaller than $\pi - \frac{3\alpha}{8h}$, because the slopes of h_{i-1} and h_{i+1} with respect to $\overline{p_{i-1}p_i}$ are $\frac{\pi}{2}$ and $\frac{\pi}{2} - \frac{3\alpha}{4h}$, respectively (notice that $\alpha \leq \frac{\pi}{4}$ and $h \geq 2$), then t_2^{i-1} intersects both h_{i-1} and h_{i+1}. It can be analogously proved

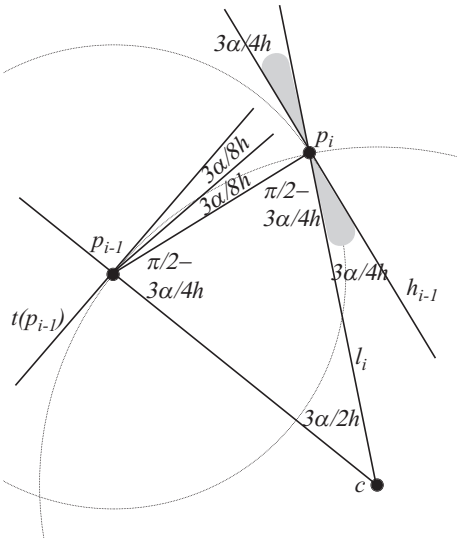


Figure 6.6: The angle between l_i and h_{i-1}.

that t_1^{i+1} intersects h_{i-1} and h_{i+1}. It follows that the intersection of H^{i-1}, H^{i+1}, $W(p_{i-1})$, and $W(p_{i+1})$ contains a triangle T as required by the claim (notice that the angle of T incident to p_i is $\frac{3\alpha}{2h}$). Considering circle C does not invalidate the existence of T, since C is concentric with C' and has a bigger radius, hence T can always be chosen sufficiently small so that it completely lies inside C.

Now Γ_i can be placed inside T, by scaling Γ_i down till it fits inside T (see Fig. 6.7(a)). The scaling always allows to place Γ_i inside T, since the angle of $R(\Gamma_i)$ incident to p_i is $\alpha_i = \frac{3\alpha}{16h}$, that is smaller than the angle of T incident to p_i, which is $\frac{3\alpha}{2h}$. In particular, we choose to place Γ_i inside T so that l_i bisects the angle of $R(\Gamma_i)$ incident to p_i. This concludes the construction of Γ.

In the following we will prove that the constructed drawing Γ satisfies Properties 1–4. However, for this purpose, we need some preliminary lemmata.

Consider the tangent $t(p^*)$ to C in p^*. Consider two half-lines l_1^* and l_2^* incident to p^*, lying in the opposite part of C with respect to $t(p^*)$, and forming angles equal to β with $t(p^*)$. Denote by $W(p^*)$ the wedge centered at p^*, delimited by l_1^* and l_2^*, and not containing C. We have the following lemmata.

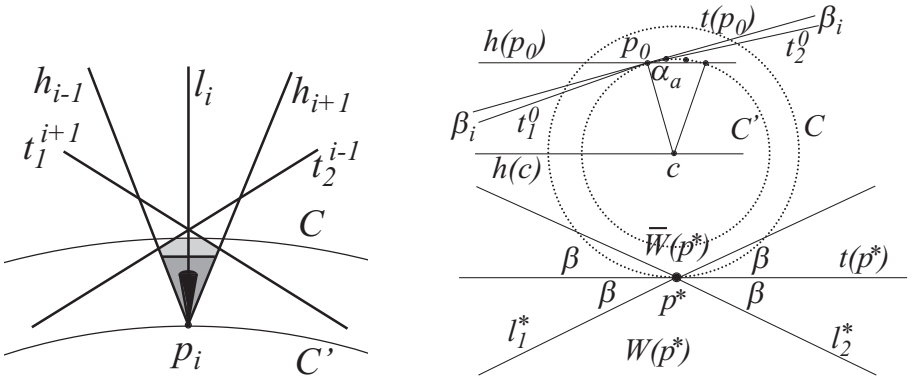


Figure 6.7: (a) Placement of Γ inside R_i. Region $R(\Gamma)$ is the darkest, triangle T is composed of $R(\Gamma)$ and of the second darkest region, R_i is composed of T and of the light shaded region. (b) Illustration for the proof of Lemma 6.1.

Lemma 6.1 *The closed wedge $W(p^*)$ is completely contained inside the open wedge $W(p_i)$, for each $i = 0, 1, \ldots, h$.*

Proof: Consider any point p_i. First, observe that p_i is contained in the wedge $\overline{W}(p^*)$ obtained by reflecting $W(p^*)$ with respect to $t(p^*)$. Namely, p_i is contained in $W(p^*, \frac{\alpha}{2})$, which is in turn contained inside $\overline{W}(p^*)$, since $\frac{\alpha}{2} < \pi - 2\beta$, as a consequence of the fact that $\frac{\pi}{4} > \beta \geq \alpha$. Hence, in order to prove the lemma, it suffices to show that the absolute value of the slope of each of t_1^i and t_2^i is smaller than the absolute value of the slope of the half-lines delimiting $W(p^*)$. Such latter half-lines form angles of β, by construction, with the x-axis.

The slope of t_1^i can be computed by adding the slope of t_1^i with respect to $t(p_i)$ and the slope of $t(p_i)$. The former slope is equal to $\beta_i = \frac{3\alpha}{8h}$, by construction. Recalling that $t(p_i)$ is the tangent to A in p_i, the slope of $t(p_i)$ is bounded by the maximum among the slopes of the tangents to points of A. Such a maximum is clearly achieved at p_0 and p_h and is equal to $\frac{3\alpha}{4}$. Namely, refer to Fig. 6.7(b) and consider the horizontal lines $h(c)$ and $h(p_0)$ through c and p_0, respectively, that are traversed by radius (c, p_0). Such a radius forms angles of $\frac{\pi}{2}$ with $t(p_0)$; hence, the slope of $t(p_0)$, that is equal to the angle between $t(p_0)$ and $h(p_0)$, is $\frac{\pi}{2}$ minus the angle α_a between $h(p_0)$ and (c, p_0).

Angle α_a is the alternate interior of the angle between $h(c)$ and (c, p_0), which is complementary to the half of angle $\widehat{p_0cp_h}$, which is equal to $\frac{3\alpha}{2}$, by construction. Hence, α_a is equal to $\frac{\pi}{2} - \frac{3\alpha}{4}$ and the slope of $t(p_0)$ is $\frac{3\alpha}{4}$.

It follows that the absolute value of the slope of t_1^i is at most $\frac{3\alpha}{4} + \frac{3\alpha}{8h}$, which is smaller than α, since $h \geq 2$, and hence smaller than β. Analogously, the absolute value of the slope of t_2^i is smaller than β, and the lemma follows. □

Corollary 6.1 *Point p^* is inside the open wedge $W(p_i)$, for each $i = 1, 2, \ldots, h$.*

Lemma 6.2 *For every pair of indices i and j such that $1 \leq i < j \leq k$, the drawing of $S(\mu_j)$ is contained inside $W(p_i)$ and the drawing of $S(\mu_i)$ is contained inside $W(p_j)$.*

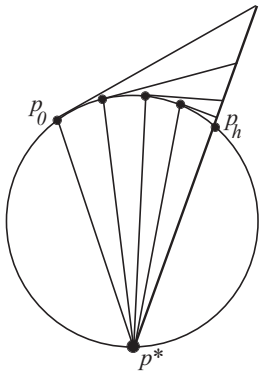


Figure 6.8: Illustration for the proof of Lemma 6.2.

Proof: We prove that the drawing of $S(\mu_j)$ is contained inside $W(p_i)$, the proof that the drawing of $S(\mu_i)$ is contained inside $W(p_j)$ being analogous. If $S(\mu_i)$ and $S(\mu_j)$ are consecutive, i.e., the cutvertices parents of $S(\mu_i)$ and $S(\mu_j)$ are u_i and u_j, with $j = i+1$, then the statement is true by construction. Suppose $S(\mu_i)$ and $S(\mu_j)$ are not consecutive. Refer to Fig. 6.8. Consider the triangle T_i delimited by (p^*, p_i), by t_2^i, and by the line through p^* and p_h. Such a triangle contains the triangle delimited by (p^*, p_{i+1}), by t_2^{i+1}, and by the line through p^* and p_h, which in turn contains the triangle delimited by (p^*, p_{i+2}), by t_2^{i+2}, and by the line through p^* and p_b'. The repetition of such an argument shows that T_i contains the triangle T_{j-1} delimited by (p^*, p_{j-1}), by t_2^{j-1}, and

by the line through p^* and p_h. By construction, Γ_j lies inside T_{j-1}, and the lemma follows. □

We are now ready to prove that the constructed drawing Γ satisfies Properties 1–4.

Property 1. We show that, for every ordered pair of vertices w_1 and w_2, there exists a distance-decreasing path from w_1 to w_2 in Γ. Observe that a distance-decreasing path from w_1 to w_2 is not necessarily a distance-decreasing path from w_2 to w_1. If both w_1 and w_2 are internal to the same graph $S(\mu_i)$, the property follows by induction. If $w_2 = r(\mu)$ and w_1 is a node in $S(\mu_i)$, then, by Property 3, there exists a path $(w_1 = v_0, v_1, \ldots, v_l = r(\mu_i))$ from w_1 to $r(\mu_i)$ such that, for every point p in $W(p_i)$, $d(v_j, p) < d(v_{j-1}, p)$, for $j = 1, 2, \ldots, l$. By Corollary 6.1, p^* is contained inside $W(p_i)$. Hence, path $(w_1 = v_0, v_1, \ldots, v_l = r(\mu_i), w_2 = r(\mu))$ is a distance-decreasing path between w_1 and w_2. If $w_1 = r(\mu)$ and w_2 is a node in $S(\mu_i)$, then, by induction, there exists a distance-decreasing path $(v_1 = r(\mu_i), v_2, \ldots, v_l = w_2)$. By Corollary 6.1, p^* is contained inside $W(p_i)$. Hence, by Property 4, $d(p_i, w_2) < d(p^*, w_2)$. It follows that path $(w_1 = r(\mu), v_1 = r(\mu_i), v_2, \ldots, v_l = w_2)$ is a distance-decreasing path between w_1 and w_2. If w_1 belongs to $S(\mu_i)$ and w_2 belongs to $S(\mu_k)$ then suppose, w.l.o.g., that $k > i$. We show the existence of a distance-decreasing path $\mathcal{P}$ in Γ, composed of three subpaths $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{P}_3$. By Property 3, Γ_i is such that there exists a path $\mathcal{P}_1 = (w_1 = v_0, v_1, \ldots, v_l = r(\mu_i))$ from w_1 to $r(\mu_i)$ such that, for every point p in $W(p_i)$, $d(v_j, p) < d(v_{j-1}, p)$, for $j = 1, 2, \ldots, l$. By Lemma 6.2, drawing Γ_k, and hence vertex w_2, is contained inside $W(p_i)$. Hence, at every vertex of path $\mathcal{P}_1$, the distance from w_2 decreases. Path $\mathcal{P}_2 = (u_i = r(\mu_i), u_{i+1}, \ldots, u_k = r(\mu_k))$ is easily shown to be distance-decreasing with respect to w_2. In fact, for each $l = i, i+1, \ldots, k-2$, angle $u_l\widehat{u_{l+1}u_k}$ is greater than $\frac{\pi}{2}$, because triangle (u_l, u_{l+1}, u_k) is inscribed in less than half a circle with u_{l+1} as middle point. Angle $u_l\widehat{u_{l+1}w_2}$ is strictly greater than $u_l\widehat{u_{l+1}u_k}$, hence it is the biggest angle in triangle (u_l, u_{l+1}, w_2), which implies $d(u_{l+1}, w_2) < d(u_l, w_2)$. By induction, there exists a distance-decreasing path $\mathcal{P}_3$ from $r(\mu_k)$ to w_2, thus obtaining a distance-decreasing path $\mathcal{P}$ from w_1 to w_2.

Property 2. Such a property holds for Γ by construction.

Property 3. Consider any node v in $S(\mu_i)$ and consider any point p internal to $W(p^*)$. By Lemma 6.1, p is internal to $W(p_i)$, as well. By induction, there

exists a path $(v = v_0, v_1, \ldots, v_l = r(\mu_i))$ such that $d(v_j, p) < d(v_{j-1}, p)$, for $j = 1, 2, \ldots, l$. Hence, path $(v = v_0, v_1, \ldots, v_l = r(\mu_i), v_{l+1} = r(\mu))$ is a path such that $d(v_j, p) < d(v_{j-1}, p)$, for $j = 1, 2, \ldots, l+1$, if and only if $d(r(\mu), p) < d(r(\mu_i), p)$. Angle $\widehat{pp^*r(\mu_i)}$ is at least $\beta + (\frac{\pi}{2} - \frac{\alpha}{2})$, which is greater than $\frac{\pi}{2}$. It follows that $(p, r(\mu_i))$ is the longest side of triangle $(p, p^*, r(\mu_i))$, thus proving that $d(p, p^*) < d(p, r(\mu_i))$ and that Property 3 holds for Γ.

Property 4. By Property 2, v is contained inside the wedge $W(p^*, \alpha)$ with angle α, centered at p^*, and bisected by the line through p^* and c. Consider any point p inside $W(p^*)$. Angle $\widehat{pp^*v}$ is at least $\beta + (\frac{\pi}{2} - \frac{\alpha}{2})$, which is greater than $\frac{\pi}{2}$. It follows that (p, v) is the longest side of triangle (p, p^*, v), thus proving that $d(p, v) < d(p^*, v)$ and that Property 4 holds for Γ.

When the induction is performed with μ equal to the root ν of the BC-tree $\mathcal{T}$, we obtain a greedy drawing of S, thus proving the following:

Theorem 6.2 *There exists an algorithm that constructs a greedy drawing of any triangulated binary cactus.*

6.4 Spanning a Triangulation with a Binary Cactus

In this section we prove the following theorem:

Theorem 6.3 *Given a triangulation G, there exists a spanning subgraph S of G such that S is a triangulated binary cactus.*

Consider any triangulation G. We are going to construct a binary cactus S spanning G. First, we outline the algorithm to construct S. Such an algorithm has several steps. At the first step, we choose a vertex u incident to the outer face of G and we construct a triangulated cycle C_T composed of u and of all its neighbors. We remove u and its incident edges from G, obtaining a biconnected internally-triangulated plane graph G^*. At the beginning of each step after the first one, we suppose to have already constructed a binary cactus S whose vertices are a subset of the vertices of G (at the beginning of the second step, S coincides with C_T), and we assume to have a set $\mathcal{G}$ of subgraphs of G (at the beginning of the second step, G^* is the only graph in $\mathcal{G}$). Each of such subgraphs is biconnected, internally-triangulated, has an outer face

whose vertices already belong to S, and has internal vertices. All such internal vertices do not belong to S and each vertex of G not belonging to S is internal to a graph in $\mathcal{G}$. Only one of the graphs in $\mathcal{G}$ may have chords. During each step, we perform the following two actions:

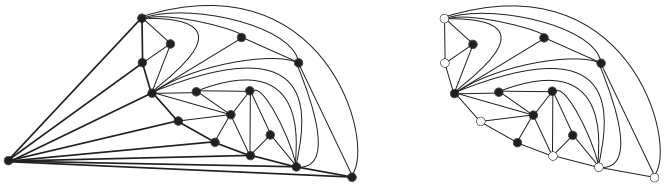
- *Action 1.* We partition the only graph G_C of $\mathcal{G}$ with chords, if any, into several biconnected internally-triangulated chordless plane graphs; we remove G_C from $\mathcal{G}$ and we add to $\mathcal{G}$ all graphs with internal vertices into which G_C has been partitioned.
- *Action 2.* We choose a graph G_i from $\mathcal{G}$, we choose a vertex u incident to the outer face of G_i and already belonging to exactly one block of S, and we add to S a block composed of u and of all its neighbors internal to G_i. We remove u and its incident edges from G_i, obtaining a biconnected internally-triangulated plane graph G_i^*. We remove G_i from $\mathcal{G}$ and we add G_i^* to $\mathcal{G}$.

The algorithm stops when $\mathcal{G}$ is empty, that is, when all the vertices of G have been spanned by S. An example of execution of the algorithm is shown in Figs. 6.9–6.16.

Now we give the details of the above outlined algorithm. At the first step of the algorithm, choose any vertex u incident to the outer face of G. Consider all the neighbors $(u_1, u_2, \ldots, u_l)$ of u in clockwise order around it. Since G is a triangulation, $C = (u, u_1, u_2, \ldots, u_l)$ is a cycle, hence the subgraph of G composed of C and of the edges connecting u to its neighbors is a triangulated cycle C_T. Let $S = C_T$. Remove vertex u and all its incident edges from G, obtaining a biconnected internally-triangulated plane graph G^*.

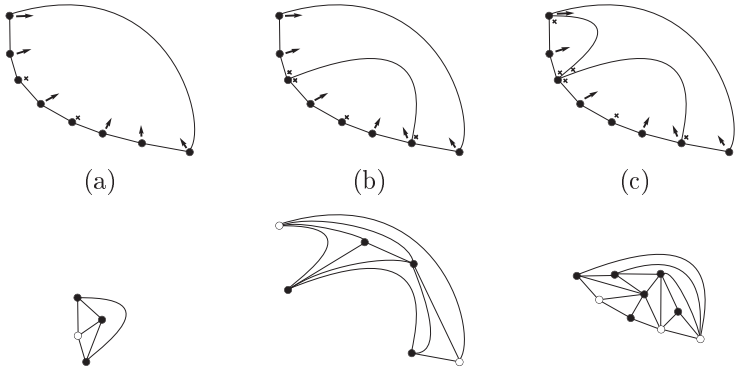
If G^* has no internal vertex, then all the vertices of G belong to S and we have the desired binary cactus spanning G. Otherwise, G^* has internal vertices. Let $\mathcal{G} = \{G^*\}$.

At each step of the algorithm, for each graph $G_i \in \mathcal{G}$, consider the vertices incident to $f(G_i)$. Each of such vertices can be either *forbidden for* G_i or *assigned to* G_i. A vertex w is forbidden for G_i if the choice of not introducing in S any new block incident to w and spanning a subgraph of G_i has been done. Conversely, a vertex w is assigned to G_i if a new block incident to w and spanning a subgraph of G_i could be introduced in S. For example, w is forbidden for G_i if two blocks of S already exist sharing w as a cutvertex. At the end of the first step of the algorithm, choose any two vertices incident to $f(G^*)$ as the only forbidden vertices for G^*. All the other vertices incident to $f(G^*)$ are assigned to G^*.



(a) (b)

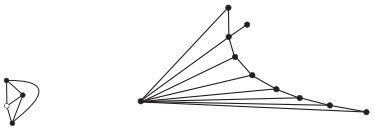
Figure 6.9: First step of the algorithm: (a) A triangulation G, from which a vertex u and its neighbors are selected. The thick subgraph is the triangulated cycle C_T such that $S = C_T$ after Step 1. (b) Graph G^* obtained from G by removing u and its incident edges. Two arbitrarily chosen vertices (represented by black circles) incident to $f(G^*)$ are forbidden for G^*, all others (represented by white circles) are assigned to it.



(a) (b) (c)

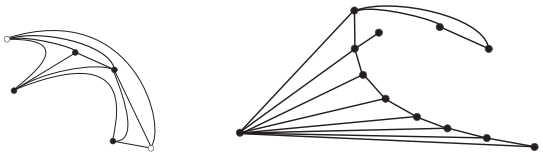
(d) (e) (f)

Figure 6.10: Step 2, Action 1. (a)–(c) Outerplane graphs O_C^0, O_C^1, and $O_C^2 = O_C$, and the assignment of vertices to their faces. (d)–(f) Graphs G_1, G_2, and G_3, where $\mathcal{G} = \{G_1, G_2, G_3\}$, obtained by partitioning G^* in biconnected, internally triangulated, chordless subgraphs.



(a) (b)

Figure 6.11: Step 2, Action 2. (a) Choice of a graph G_i in $\mathcal{G}$ (here $G_i = G_1$) and of a vertex u incident to $f(G_i)$. The thick subgraph is the edge (u, u_1) added to S after Action 2 of Step 2. (b) Binary cactus S after Action 2 of Step 2. $\mathcal{G} = \{G_2, G_3\}$.

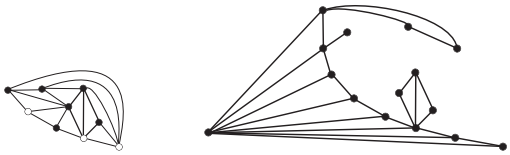


(a) (b)

Figure 6.12: Step 3, Action 2 (Action 1 of Step 3 is skipped because no graph in $\mathcal{G}$ has chords). (a) Choice of a graph G_i in $\mathcal{G}$ (here $G_i = G_2$) and of a vertex u incident to $f(G_i)$. The thick subgraph is the triangulated cycle C_T added to S after Step 3, Action 2. (b) Binary cactus after Action 2 of Step 3. $\mathcal{G} = \{G_3\}$.

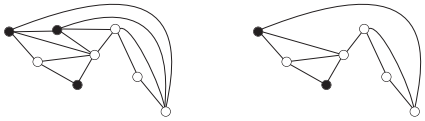
At the beginning of the i-th step, with $i \geq 2$, we assume that each of the following holds:

- *Invariant A:* Graph S is a binary cactus spanning all and only the vertices that are not internal to any graph in $\mathcal{G}$.
- *Invariant B:* Each graph in $\mathcal{G}$ is biconnected, internally-triangulated, and has internal vertices.
- *Invariant C:* Only one of the graphs in $\mathcal{G}$ may have chords.
- *Invariant D:* No internal vertex of a graph $G_i \in \mathcal{G}$ belongs to a graph $G_j \in \mathcal{G}$, with $i \neq j$.
- *Invariant E:* For each graph $G_i \in \mathcal{G}$, all the vertices incident to $f(G_i)$ are assigned to G_i, except for two vertices, which are forbidden.



(a) (b)

Figure 6.13: Step 4, Action 2 (Action 1 of Step 4 is skipped because no graph in $\mathcal{G}$ has chords). (a) Choice of a graph G_i in $\mathcal{G}$ (here $G_i = G_3$) and of a vertex u incident to $f(G_i)$. The thick subgraph is the triangulated cycle C_T added to S after Step 4, Action 2. (b) Binary cactus S after Action 2 of Step 4. $\mathcal{G} = \{G_3^*\}$, where G_3^* is the graph obtained from G_3 by removing u and its incident edges.



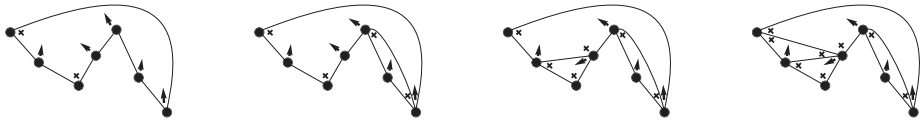
(a) (b)

Figure 6.14: Step 5, before Action 1. (a) The only graph $G_C = G_3^*$ in $\mathcal{G}$, with its assigned vertices (white circles) and forbidden vertices (black circles). (b) The outerplane graph O_C induced by the vertices incident to $f(G_C)$.

- *Invariant F:* Each vertex v incident to the outer face of a graph in $\mathcal{G}$ is assigned to at most one graph $G_i \in \mathcal{G}$. If a vertex v incident to the outer face of a graph in $\mathcal{G}$ is assigned to a graph $G_i \in \mathcal{G}$, then v is forbidden for all graphs $G_j \in \mathcal{G}$ such that v is incident to $f(G_j)$, with $j \neq i$.
- *Invariant G:* Each vertex assigned to a graph in $\mathcal{G}$ belongs to exactly one block of S.

Such invariants clearly hold after the first step of the algorithm. During each step of the algorithm after the first one, we perform the following two actions.

Action 1: If $\mathcal{G}$ does not contain any graph with chords, go to Action 2. Otherwise, by Invariant C, only one of the graphs in $\mathcal{G}$, say G_C, has chords. We use the chords of G_C to partition it into k biconnected, internally-triangulated, chordless graphs G_C^j, with $j = 1, 2, \ldots, k$.



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

Figure 6.15: Step 5, Action 1. (a)–(d) Outerplane graphs O_C^0, O_C^1, O_C^2, $O_C^3 = O_C$, and the assignment of vertices to their faces. Partitioning G_C into subgraphs G_C^j produces only one graph, say G_4, with internal vertices. Hence, set $\mathcal{G}$ is now $\{G_4\}$.

Consider the subgraph O_C of G_C induced by the vertices incident to $f(G_C)$. Clearly, O_C is a biconnected outerplane graph. To each internal face f of O_C delimited by a cycle C, a graph G_C^j is associated such that G_C^j is the subgraph of G_C induced by the vertices of C or inside C. We are going to replace G_C with graphs G_C^j in $\mathcal{G}$. However, we first show how to decide which vertices incident to the outer face of a graph G_C^j are assigned to G_C^j and which vertices are forbidden for G_C^j. Since each graph G_C^j is univocally associated with a face of O_C, in the following we assign vertices to the faces of O_C and we forbid vertices for the faces of O_C, meaning that if a vertex is assigned to (forbidden for) a face f of O_C, then it is assigned to (resp. forbidden for) the associated graph G_C^j.

We want to assign the vertices incident to $f(O_C)$ to faces of O_C so that:

- Property 1: No forbidden vertex is assigned to any face of O_C;
- Property 2: No vertex is assigned to more than one face of O_C;
- Property 3: Each face of O_C has exactly two incident vertices which are forbidden for it; all the other vertices of the face are assigned to it.

By Invariant E, G_C has two forbidden vertices. We construct an assignment of vertices to the faces of O_C in some steps. Let p be the number of chords of O_C. Consider the Hamiltonian cycle O_C^0 of O_C, and assign all the vertices of O_C^0, but for the two forbidden vertices, to the only internal face of O_C^0. At the i-th step, $1 \leq i \leq p$, we insert into O_C^{i-1} a chord of O_C, obtaining a graph O_C^i. This is done so that Properties 1–3 are satisfied by O_C^i (with O_C^i instead of O_C). After all the p chords of O_C have been inserted, $O_C^p = O_C$, and we have an assignment of vertices to faces of O_C satisfying Properties 1–3.

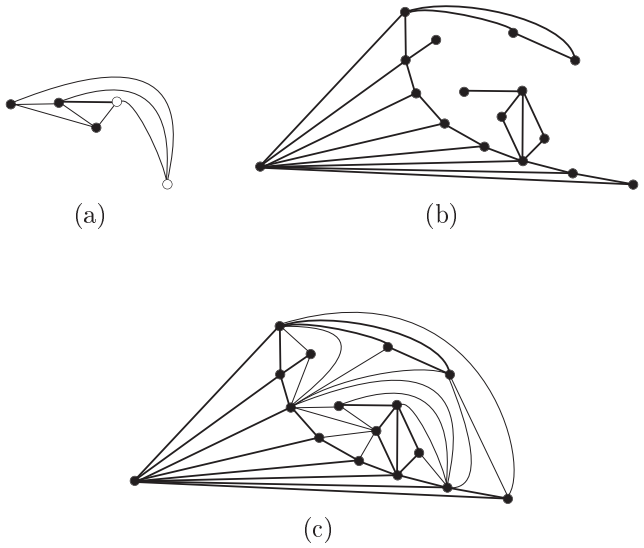


Figure 6.16: Step 5, Action 2: (a) Choice of a graph G_i in $\mathcal{G}$ (here $G_i = G_4$) and of a vertex u incident to $f(G_i)$. The thick subgraph is the edge (u, u_1) added to S after Step 5, Action 2. (b) Binary cactus S at the end of the algorithm. (c) The obtained binary cactus S spans G.

Properties 1–3 are clearly satisfied by the assignment of vertices to the faces of O_C^0. Inductively assume that Properties 1–3 are satisfied by the assignment of vertices to the faces of O_C^{i-1}. Let (u_a, u_b) be the chord that is inserted at the i-th step. Chord (u_a, u_b) partitions a face f of O_C^{i-1} into two faces f_1 and f_2. By Property 3, two vertices u_1^* and u_2^* incident to f are forbidden for it and all other vertices incident to f are assigned to it. For each face of O_C^i different from f_1 and f_2, assign and forbid vertices as in the same face in O_C^{i-1}. Assign and forbid vertices for f_1 and f_2 as follows:

- If vertices u_a and u_b are the same vertices as u_1^* and u_2^* (see Fig. 6.17), assign to f_1 and f_2 all the vertices incident to it, except for u_a and u_b. No forbidden vertex has been assigned to any face of O_C^i (Property 1). Vertices u_a and u_b have not been assigned to any face. All the vertices assigned to f belong to exactly one of f_1 and f_2 and so they have been assigned to exactly one face (Property 2). The only vertices of f_1 (resp.

of f_2) not assigned to it are u_a and u_b, while all the other vertices are assigned to such a face (Property 3).

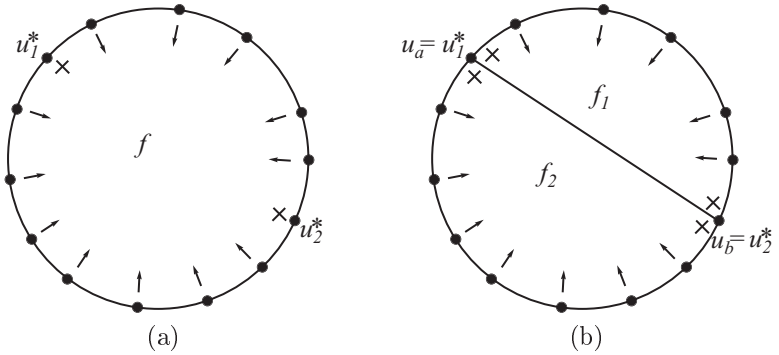
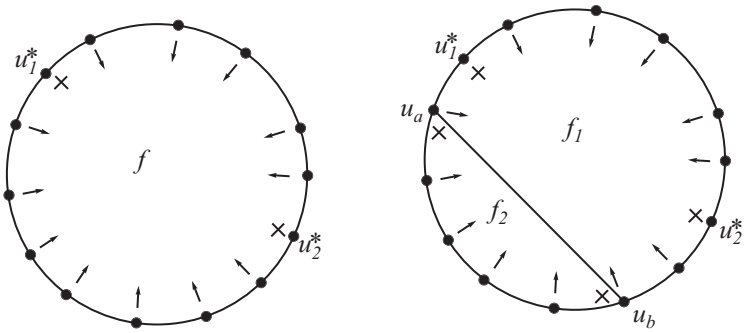


Figure 6.17: Vertices u_a and u_b are the same vertices of u_1^* and u_2^*.

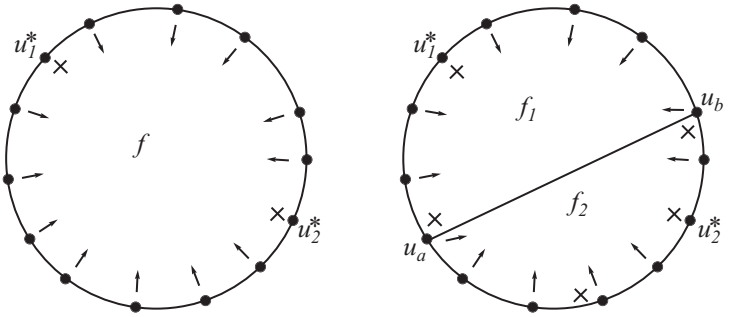
- If vertices u_a and u_b are both distinct from each of u_1^* and u_2^* and both u_1^* and u_2^* are in the same of f_1 and f_2, say in f_1 (see Fig. 6.18), assign to f_1 all the vertices incident to it, except for u_1^* and u_2^*, and assign to f_2 all the vertices incident to it, except for u_a and u_b. No forbidden vertex has been assigned to any face of O_C^i (Property 1). Vertices u_a and u_b have been assigned to exactly one face, namely f_1. All the other vertices assigned to f belong to exactly one of f_1 and f_2 and so they have been assigned to exactly one face (Property 2). The only vertices of f_1 (resp. of f_2) not assigned to it are u_1^* and u_2^* (resp. u_a and u_b), while all the other vertices are assigned to such a face (Property 3).

- If vertices u_a and u_b are both distinct from each of u_1^* and u_2^* and one of u_1^* and u_2^*, say u_1^*, is in f_1 while u_2^* is in f_2 (see Fig. 6.19), assign to f_1 all the vertices incident to it, except for u_1^* and u_a, and assign to f_2 all the vertices incident to it, except for u_2^* and u_b. No forbidden vertex has been assigned to any face of O_C^i (Property 1). Vertices u_a and u_b have been assigned to exactly one face, namely f_2 and f_1, respectively. All the other vertices assigned to f belong to exactly one of f_1 and f_2 and so they have been assigned to exactly one face (Property 2). The only vertices of f_1 (resp. of f_2) not assigned to it are u_1^* and u_a (resp. u_2^* and u_b), while all the other vertices are assigned to such a face (Property 3).



(a) (b)

Figure 6.18: Vertices u_a and u_b are both distinct from each of u_1^* and u_2^* and both u_1^* and u_2^* are in f_1.



(a) (b)

Figure 6.19: Vertices u_a and u_b are both distinct from each of u_1^* and u_2^*, u_1^* is in f_1, and u_2^* is in f_2.

- If one of the vertices u_1^* and u_2^* coincides with one of u_a and u_b, say u_1^* coincides with u_a, and u_2^* is in one of f_1 and f_2, say in f_1 (see Fig. 6.20), assign to f_1 all the vertices incident to it, except for u_2^* and u_a, and assign to f_2 all the vertices incident to it, except for u_a and u_b. No forbidden vertex has been assigned to any face of O_C^i (Property 1). Vertex u_a has not been assigned to any face and vertex u_b has been assigned to exactly one face, namely f_1. All the other vertices assigned to f belong to exactly one of f_1 and f_2 and so they have been assigned to exactly one

face (Property 2). The only vertices of f_1 (resp. of f_2) not assigned to it are u_2^* and u_a (resp. u_a and u_b), while all the other vertices are assigned to such a face (Property 3).

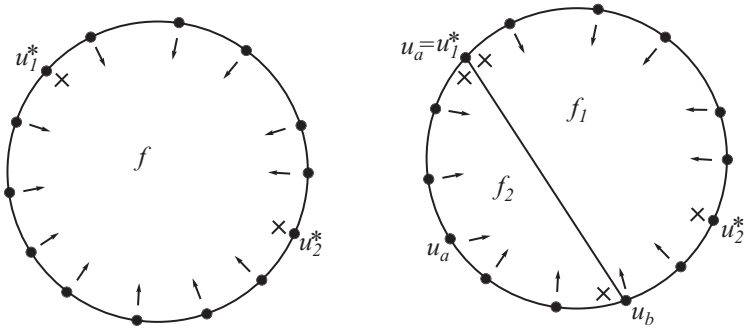


Figure 6.20: Vertex u_1^* coincides with u_a and vertex u_2^* is in f_1.

Graph G_C is removed from $\mathcal{G}$. All the graphs G_C^j having internal vertices are added to $\mathcal{G}$. We prove that Invariants A–G are satisfied after Action 1.

Invariant A: A vertex is internal to a graph in $\mathcal{G}$ after Action 1 if and only if it is internal to a graph in $\mathcal{G}$ before Action 1. Since no block is added to S during Action 1, then Invariant A holds after Action 1.

Invariant B: By construction, each graph G_C^j inserted into $\mathcal{G}$ after Action 1 has internal vertices. Further, G_C^j is the graph contained inside a simple cycle of a biconnected internally triangulated plane graph, hence it is biconnected and internally triangulated, as well, satisfying Invariant B.

Invariant C: By Invariant C, before Action 1 only graph G_C may have chords among the graphs in $\mathcal{G}$. After Action 1, however, G_C is replaced in $\mathcal{G}$ by chordless graphs and hence no graph in $\mathcal{G}$ has chords, satisfying Invariant C.

Invariant D: By Invariant D, each vertex that, before Action 1, is internal to a graph $G_i \neq G_C$ in $\mathcal{G}$ does not belong to any graph $G_j \neq G_i$ in $\mathcal{G}$. Since the set of vertices belonging to graphs G_C^j is a subset of the vertices of G_C, after Action 1 Invariant D holds for all the vertices internal to a

graph $G_i \neq G_C^j$. An internal vertex of a graph G_C^j is an internal vertex of G_C, as well, hence, by Invariant D, it does not belong to any graph that has not been introduced in $\mathcal{G}$ during Action 1. It remains to prove that an internal vertex of a graph G_C^j does not belong to any graph G_C^l, with $l \neq j$. By construction, the internal vertices of such graphs are inside cycles corresponding to distinct faces of O_C. Hence, an internal vertex of G_C^j does not belong to G_C^l.

Invariant E: Invariant E holds for all the graphs that are in $\mathcal{G}$ before Action 1 and that are still in $\mathcal{G}$ after Action 1. By Property 3, each graph G_C^j inserted into $\mathcal{G}$ after Action 1 satisfies Invariant E.

Invariant F: All the vertices that, before Action 1, are assigned to a graph $G_i \neq G_C$ in $\mathcal{G}$ satisfy Invariant F after Action 1. Namely, by Invariant F before Action 1, if they are incident to $f(G_C)$, then they are forbidden for G_C and, by Property 1, they are not assigned to any graph G_C^j. By Invariant F, before Action 1 each vertex w assigned to G_C is not assigned to any graph $G_i \neq G_C$ in $\mathcal{G}$. After Action 1, G_C is not a graph in $\mathcal{G}$ any longer, hence w is not assigned to it. By Property 2, after Action 1 each vertex is assigned to at most one graph G_C^j, hence Invariant F holds after Action 1.

Invariant G: Since no block is added to S during Action 1, and since the set of vertices assigned to graphs in $\mathcal{G}$ after Action 1 is a subset of the set of vertices assigned to graphs in $\mathcal{G}$ before Action 1, then Invariant G holds after Action 1.

Action 2: After Action 1 all graphs in $\mathcal{G}$ are chordless. Notice that there is at least one graph G_i in $\mathcal{G}$, otherwise the algorithm would have stopped before Action 1. By Invariant B, G_i has internal vertices. Choose any vertex u that is incident to $f(G_i)$ and that is assigned to G_i (see Fig. 6.21). By the biconnectivity of G_i and by the fact that it has internal vertices, $f(G_i)$ has at least three vertices. Since each graph in $\mathcal{G}$ has at most two forbidden vertices (by Invariant E), a vertex u assigned to G_i always exists. Consider all the neighbors $(u_1, u_2, \ldots, u_l)$ of u internal to G_i, in clockwise order around u. Since G is biconnected, chordless, internally triangulated, and has internal vertices, then $l \geq 1$. If $l = 1$, then let C_T be edge (u, u_1). Otherwise, let C_T be the triangulated cycle composed of cycle $(u, u_1, u_2, \ldots, u_l)$ and of the edges connecting u to its neighbors. Add C_T to S. Remove u and its incident edges from G_i, obtaining a graph G_i^*. Assign to G_i^* all the vertices incident to $f(G_i^*)$,

except for the two vertices that are forbidden for G_i. Remove G_i from $\mathcal{G}$ and insert G_i^*, if it has internal vertices, into $\mathcal{G}$.

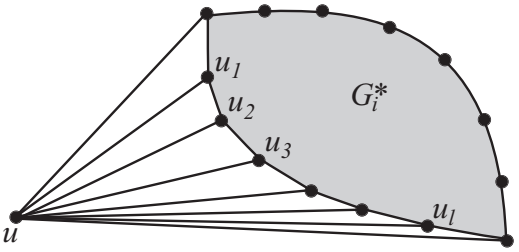


Figure 6.21: Action 2 of a step of the algorithm.

We prove that Invariants A–G are satisfied after Action 2.

Invariant A: The block $(u, u_1, u_2, \ldots, u_l)$ added to S is either an edge or a triangulated cycle. By Invariant A, before Action 2 all vertices internal to a graph in $\mathcal{G}$ are not spanned by S. Further, by Invariant G, before Action 2 vertex u belongs to exactly one block of S. It follows that S is still a binary cactus after Action 2. Before Action 2, S spans all and only the vertices that are not internal to any graph in $\mathcal{G}$. The only vertices that are internal to a graph in $\mathcal{G}$ before Action 2 and that are incident to the outer face of a graph in $\mathcal{G}$ after Action 2 are $u_1, u_2, \ldots, u_l$, which are spanned by S after Action 2. Hence, S spans all the vertices of G that are not internal to any graph in $\mathcal{G}$. Before Action 2, no internal vertex of a graph in G is spanned by S. The vertices which are added to S during Action 2 are incident to $f(G_i^*)$, hence, by Invariant D to be proved below, they are not internal to any graph in $\mathcal{G}$ after Action 2. Hence, S does not span vertices of G that are internal to a graph in $\mathcal{G}$, satisfying Invariant A.

Invariant B: By construction, G_i^* is the only graph inserted into $\mathcal{G}$ after Action 2. However, G_i^* is biconnected and internally triangulated, since it is obtained from a graph G_i that, by Invariant B before Action 2, is biconnected, internally triangulated, chordless, and has internal vertices, by removing a vertex incident to $f(G_i)$. Further, G_i^* has internal vertices, otherwise it would not have been inserted into $\mathcal{G}$. Hence, Invariant B is satisfied after Action 2.

Invariant C: Before Action 2, all graphs in $\mathcal{G}$ have no chord. At most one graph, namely G_i^*, is inserted into $\mathcal{G}$ after Action 2, hence Invariant C is still satisfied.

Invariant D: By Invariant D, before Action 2 no internal vertex of a graph $G_l \neq G_i$ in $\mathcal{G}$ belongs to a graph $G_j \neq G_l$ in $\mathcal{G}$. Since the vertices of G_i^* are a subset of the vertices of G_i then, after Action 2, Invariant D holds for each internal vertex of G_l. Further, the internal vertices of G_i^* are a subset of the internal vertices of G_i and hence, after Action 2, Invariant D holds also for each internal vertex of G_i^*.

Invariant E: Invariant E holds for all the graphs that are in $\mathcal{G}$ before Action 2 and that are still in $\mathcal{G}$ after Action 2. By construction, all the vertices incident to the outer face of G_i^*, except for the two forbidden vertices of G_i, are assigned to G_i^*, satisfying Invariant E.

Invariant F: The only vertices that are assigned to a graph in $\mathcal{G}$ during Action 2 are the vertices incident to the outer face of G_i^*. All the vertices internal to G_i before Action 2 and incident to the outer face of G_i^* after Action 2 are assigned exclusively to G_i^*, namely if before Action 2 one of such vertices is assigned to a graph $G_j \neq G_i$, then such a vertex would be incident to the outer face of G_j, contradicting Invariant D. All the vertices that are assigned to G_i before Action 2 and that are incident to the outer face of G_i^* after Action 2, are assigned exclusively to G_i before Action 2, by Invariant F, and hence they are assigned only to G_i^* after Action 2. All the vertices that are assigned to a graph different from G_i are such that, if they are incident to the outer face of G_i, then they are forbidden for it. Since all the vertices forbidden for G_i are forbidden for G_i^*, then Invariant F holds for such vertices, as well.

Invariant G: The block added to S after Action 2 spans only vertices internal to G_i and vertex u. Hence, all the vertices assigned to a graph in $\mathcal{G}$ and not belonging to G_i are still spanned by a single block of S. All the vertices incident to the outer face of G_i, except for u, are not spanned by the block added during Action 2. All the vertices internal to G_i and assigned to G_i^* are spanned by the only block added during Action 2. Finally, after Action 2, vertex u is not assigned to any graph in $\mathcal{G}$ any longer.

When the algorithm stops, i.e., when there is no graph in $\mathcal{G}$, by Invariant A graph S is a binary cactus spanning all vertices of G, hence proving Theorem 6.3.

6.5 Extension to Triconnected Planar Graphs

In this section, we show how slight modifications of the two main arguments (see Sect. 6.3 and Sect. 6.4) used to prove that every triangulation has a greedy drawing allow us to prove Conjecture 6.1. First, we show how to construct a greedy drawing of any *non-triangulated binary cactus*, that is a connected graph such that: (i) the block associated with each B-node of $\mathcal{T}$ is either an edge or a simple cycle; and (ii) every cutvertex is shared by exactly two blocks of S. Second, we show that a triconnected planar graph can always be spanned by a non-triangulated binary cactus. Notice that a non-triangulated binary cactus is easily obtained from a triangulated binary cactus by removing the edges internal to the triangulated cycles.

It is not difficult to argue that the algorithm shown in Sect. 6.3 also constructs greedy drawings of any non-triangulated binary cactus S. More specifically, construct the BC-tree $\mathcal{T}$ of S; consider each block $(r(\mu) = u_0, u_1, \ldots, u_{h-1})$ corresponding to a B-node μ of $\mathcal{T}$ and insert a dummy edge between $r(\mu)$ and each node u_i, with $1 \leq i \leq h-2$; the resulting graph S' is a triangulated binary cactus; apply the algorithm described in Sect. 6.3 to construct a greedy drawing Γ' of S'; finally, remove dummy edges from Γ', obtaining a drawing Γ of S.

We claim that Γ is a greedy drawing. Notice that the validity of Lemmata 6.1 and 6.2 only depends on the angles of the geometric construction. Hence, such Lemmata hold for Γ. Then, it is sufficient to prove that at each step of the induction Γ satisfies Properties 1–4 described in Sect. 6.3.

Actually, Property 2 and Property 4 are trivially verified, since they only depend on the angles of the construction.

The proof of Property 1 can be conduced analogously to the one presented in Sect. 6.3, namely by proving that, for every pair of vertices w_1 and w_2, there exists a distance-decreasing path between them. However, the case in which the distance-decreasing path contains edge $(u_i, r(\mu))$, for some $2 \leq i \leq h-2$, deserves an explicit discussion, because such an edge is no longer an edge of the graph. Observe that it can be supposed that one out of w_1 and w_2 is $r(\mu)$, because in all the other cases the distance-decreasing path between w_1 and w_2 does not contain $(u_i, r(\mu))$.

First, suppose that the path ends at $r(\mu)$, i.e., $w_2 = r(\mu)$. Edge $(u_i, r(\mu))$ can be replaced either by path $(u_i, u_{i-1}, \ldots, u_1, u_0)$ or by path $(u_i, u_{i+1}, \ldots, u_{h-1}, u_0)$, depending on whether $i \leq \frac{h}{2}$ or $i \geq \frac{h}{2}$, still leaving the path distance-decreasing. In fact (see Fig.6.22(a)), denote by p' the intersection point between C' and segment $\overline{cp^*}$ and suppose that $i \geq \frac{h}{2}$, the case in which $i \leq \frac{h}{2}$ being analogous; angle $\widehat{u_i u_{i+1} p'}$ is greater than or equal to $\frac{\pi}{2}$ because triangle (u_i, u_{i+1}, p') is inscribed in no more than half a circle with u_{i+1} as middle point; then, angle $\widehat{u_i u_{i+1} p^*}$ is also greater than $\frac{\pi}{2}$ because it is strictly greater than $\widehat{u_i u_{i+1} p'}$; hence, $\overline{p^* u_i}$ is longer than $\overline{p^* u_{i+1}}$; it follows that, when traversing edge (u_i, u_{i+1}), the path decreases its distance from the point p^* where $r(\mu)$ is drawn.

Second, suppose that the path starts at $r(\mu)$, i.e., $w_1 = r(\mu)$. Edge $(r(\mu), u_i)$ can be replaced either by path $(u_0, u_1, \ldots, u_{i-1}, u_i)$ or by path $(u_0, u_{h-1}, \ldots, u_{i+1}, u_i)$, depending on whether $i \leq \frac{h}{2}$ or $i \geq \frac{h}{2}$, still leaving the path distance-decreasing. In fact, suppose that $i \geq \frac{h}{2}$, the case in which $i \leq \frac{h}{2}$ being analogous; as in the previous case, edge $(r(\mu), u_{h-1})$ can be shown to decrease the distance from w_2 by considering triangle $(r(\mu), u_{h-1}, w_2)$ and arguing that angle $\widehat{p^* u_{h-1} w_2}$ is greater than $\frac{\pi}{2}$. Further, path $(u_{h-1}, u_{h-2}, \ldots, u_{i+1}, u_i, \ldots, w_2)$ can be shown to be distance-decreasing as in the proof of Property 1 in Sect. 6.3 (in the case in which w_1 belongs to $S(\mu_i)$ and w_2 belongs to $S(\mu_j)$).

In order to prove Property 3, it is sufficient to observe that an edge $(u_i, r(\mu))$ can be replaced either by path $(u_i, u_{i-1}, \ldots, u_1, u_0)$ or by path $(u_i, u_{i+1}, \ldots, u_{h-1}, u_0)$, still obtaining a path in which at every step the distance from any point in $W(p^*)$ decreases. In fact (see Fig.6.22 (b)), denote by p any point inside $W(p^*)$, and denote by $a_{i-1,i}$ and $a_{i,i+1}$ the axes of segments $\overline{p_{i-1} p_i}$ and $\overline{p_i p_{i+1}}$, respectively. Since $a_{i-1,i}$ and $a_{i,i+1}$ intersect in the center of C', we have that p is either to the left of $a_{i-1,i}$ or to the right of $a_{i,i+1}$, or both. Suppose that p is to the right of $a_{i,i+1}$, the other case being analogous. Then, $d(p, p_{i+1}) < d(p, p_i)$. The repetition of such an argument leads to prove that path $(u_i, u_{i+1}, \ldots, u_{h-1}, u_0)$ decreases the distance from p at every vertex.

As we proved that there exists an algorithm to construct greedy drawings of non-triangulated binary cactuses, in order to prove Conjecture 6.1 it suffices to show that every triconnected planar graph admits a non-triangulated binary cactus as a spanning subgraph. In the following we sketch how to extend the arguments of Sect. 6.4 in order to prove such a result.

The algorithm to find a non-triangulated binary cactus spanning a given triconnected planar graph G consists of several steps, in which the cactus is constructed incrementally by adding to it one block at a time. As in the triangulated case, at the beginning of each step after the first one, we suppose

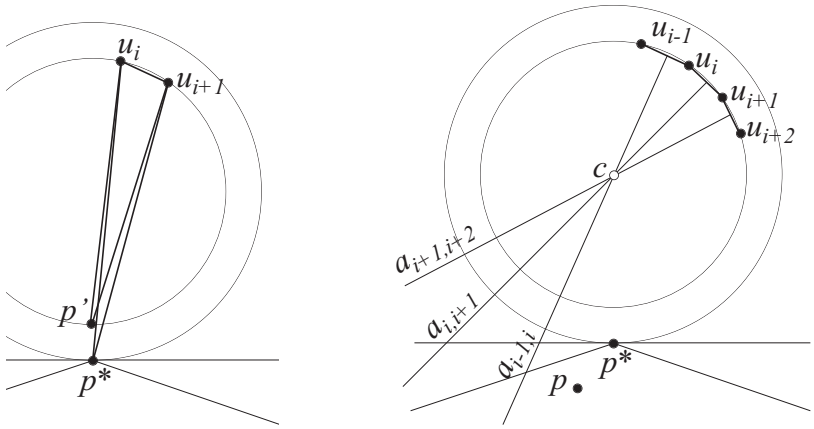


Figure 6.22: (a) When traversing edge (u_i, u_{i+1}), the distance from p^* decreases. (b) When traversing edge (u_i, u_{i+1}), the distance from p decreases.

to have already constructed a non-triangulated binary cactus S whose vertices are a subset of the vertices of G, and we assume to have a set $\mathcal{G}$ of subgraphs of G. Further, we assume that the following invariants hold:

- *Invariant A:* Graph S is a non-triangulated binary cactus spanning all and only the vertices that are not internal to any graph in $\mathcal{G}$.
- *Invariant B:* Each graph in $\mathcal{G}$ is biconnected and has internal vertices.
- *Invariant C:* At most one graph $G_C \in \mathcal{G}$ has separation pairs. However, if G_C exists, each of its separation pairs has both vertices incident to $f(G_C)$.
- *Invariant D:* No internal vertex of a graph $G_i \in \mathcal{G}$ belongs to a graph $G_j \in \mathcal{G}$, with $i \neq j$.
- *Invariant E:* For each graph $G_i \in \mathcal{G}$, all the vertices incident to $f(G_i)$ are assigned to G_i, except for two vertices, which are forbidden.
- *Invariant F:* Each vertex v incident to the outer face of a graph in $\mathcal{G}$ is assigned to at most one graph $G_i \in \mathcal{G}$. If a vertex v incident to the outer

face of a graph in $\mathcal{G}$ is assigned to a graph $G_i \in \mathcal{G}$, then v is forbidden for all graphs $G_j \in \mathcal{G}$ such that v is incident to $f(G_j)$, with $j \neq i$.

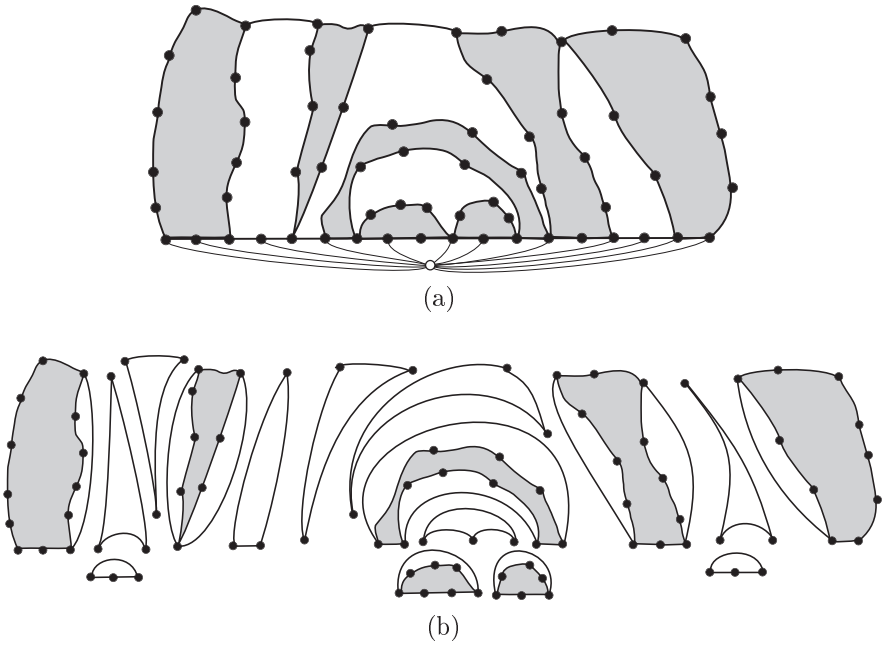
- *Invariant G:* Each vertex assigned to a graph in $\mathcal{G}$ belongs to exactly one block of S.

During each step, we perform two different actions. Action 1 removes from $\mathcal{G}$ the only graph G_C which contains separation pairs, if such a graph exists, and partitions G_C into a set of triconnected planar graphs G_C^i to be added to $\mathcal{G}$. Action 2 removes from a graph $G_i \in \mathcal{G}$ a vertex incident to $f(G_i)$ and its incident edges and creates a new block to be added to S. At the end of each of the two actions, Invariants A–G are satisfied. The algorithm stops when $\mathcal{G}$ is empty, that is, when all the vertices of G are spanned by S.

A first difference between the triangulated and the non-triangulated case concerns the first step of the algorithm. Namely, while in the triangulated case we select one vertex v of the outer face and we initialize the cactus with the block composed of v and of its neighbors, in this new algorithm we initialize the cactus with the cycle delimiting the outer face.

Another important difference lies in Action 1, that is, in the way the graph G_C which may be not triconnected is partitioned into subgraphs. In the triangulated case, such a partition is done by considering the chords of $f(G_C)$. In the non-triangulated case we have to more generally consider separation pairs incident to $f(G_C)$, since we are not guaranteed that every two vertices composing a separation pair are joined by an edge. Refer to Fig. 6.23. The partition is performed by considering one separation pair at a time. At the beginning of every step of such an algorithm, we have a partition of G_C into a set of graphs G_i. Each graph G_i which still has a separation pair is further partitioned into two subgraphs G_i^1 and G_i^2 and each of the vertices incident to $f(G_i)$ is assigned to, or forbidden for, G_i^1 and G_i^2 by means of the same algorithm described in Sect. 6.4. Hence, the assignment of the vertices to the graphs G_i^1 and G_i^2 can be done in such a way that the invariant that each of G_i^1 and G_i^2 has at most two forbidden vertices is maintained. A dummy edge connecting the two vertices of the separation pair has to be added incident to the outer face of each of G_i^1 and G_i^2, if it does not exist yet, in order to maintain the invariant that all the vertices of the outer faces of G_i^1 and G_i^2 have already been assigned to some block of S. Such a dummy edge is incident to the outer faces of G_i^1 and G_i^2 and hence it will not be part of any new block that is added to S in the following steps of the algorithm. It is easy to see that the described procedure for partitioning a graph into subgraphs does not

introduce new separation pairs, does not introduce multiple edges, and hence it terminates providing a set of triconnected planar graphs.



(a)

(b)

Figure 6.23: Partition of a biconnected graph having all separating pairs incident to the outer face into a set of triconnected planar graphs.

Concerning Action 2, while in the triangulated case at every step we add to the cactus either an edge or a triangulated cycle, in the non-triangulated case we add either an edge or a simple cycle. Such a cycle is obtained as follows (see Fig. 6.24). As in the triangulated case, consider a vertex v incident to the outer face of a subgraph $G_i \in \mathcal{G}$ and such that v is assigned to G_i. Consider the internal faces of G_i that are incident to v, except for the two faces f_1 and f_2 sharing an edge with $f(G_i)$. Add to S the cycle that passes through all the vertices that are incident to such faces. Remove vertex v and its incident edges from G_i, obtaining a new graph G_i^*. Consider the two vertices v_1' and v_1'' adjacent to v and belonging to f_1. A dummy edge (v_1', v_1'') is added to G_1^*, if it does not exist yet, incident to $f(G_i^*)$. Analogously, consider the two vertices

v_2' and v_2'' adjacent to v and belonging to f_2 and add a dummy edge (v_2', v_2'') to G_i^*, if it does not exist yet, incident to $f(G_i^*)$. Such dummy edges allow to maintain the invariant that all the vertices incident to $f(G_i^*)$ have already been assigned to some block of S.

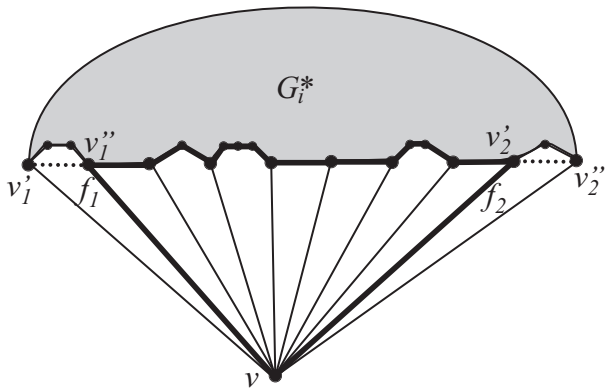


Figure 6.24: Action 2. The thick cycle is added to S. The dotted edges are inserted incident to $f(G_i^*)$, in order to maintain the invariant that all the vertices incident to $f(G_i^*)$ have already been assigned to some block of S.

We choose to present the algorithm for triangulations as the main contribution of this chapter because a proof of Conjecture 6.1 was very recently and independently presented by Leighton and Moitra at FOCS'08 [LM08]. Surprisingly, the approach used by Leighton and Moitra is exactly the same as ours. In fact, in [LM08] the authors define a class of graphs, called *Christmas cactus graphs*, which coincides with the class of non-triangulated binary cactuses; they show an algorithm to construct greedy drawings of Christmas cactus graphs and they show that every triconnected planar graph is spanned by a Christmas cactus graph. However, the way such results are achieved differs from ours. Such an issue is discussed below.

Concerning the geometric construction of greedy drawings of Christmas cactus graphs, the algorithm by Leighton and Moitra is quite similar to ours, even if a slightly different construction is used. Their algorithm places the nodes of the graph on a set of concentric circles $C_0, C_1, \ldots, C_k$, so that the block corresponding to the root ν of the BC-tree $\mathcal{T}$ has its nodes placed on C_0 and each block μ at depth i (where the depth is meant to be the number of B-nodes in the path from ν to μ in $\mathcal{T}$) is placed on C_i, except for the C-

node parent of μ, which is placed on C_{i-1}. The difference between the radii of two consecutive circles (and hence the length of the edges of the drawing) exponentially decreases with i.

Concerning the construction of a Christmas cactus graph spanning a given triconnected planar graph, we have the main differences between our techniques and Leighton and Moitra's ones. In fact, in order to show that every triconnected planar graph is spanned by a Christmas cactus graph, they use some results from a paper [GR94] by Gao and Richter.

Define a *circuit graph* to be an ordered pair (G, C) such that: (1) G is 2-connected and C is a polygon in G; (2) there exists an embedding of G in the plane such that C bounds a face; and (3) every separating pair of G has both vertices belonging to C. Hence, circuit graphs are a superclass of triconnected planar graphs. Define a *chain of blocks* $B_{i,1}, b_{i,1}, B_{i,2}, b_{i,2}, \ldots, B_{i,k_i-1}, b_{i,k_i-1}, B_{i,k_i}$ to be a connected graph such that each block contains at most two cutvertices and each cutvertex is shared by exactly two blocks.

In [GR94], Gao and Richter prove some strong structural results about circuit graphs, which are briefly described below. Gao and Richter prove that, given a circuit graph (G, C) and given two vertices x and y belonging to C, there exists a partition of $V(G)-V(C)$ into subsets $V_1, V_2, \ldots, V_m$ and there exist distinct vertices $v_1, v_2, \ldots, v_m \in V(C)-\{x, y\}$ such that: (i) the subgraph induced by $V_i \cup \{v_i\}$ is a chain of blocks $B_{i,1}, b_{i,1}, B_{i,2}, b_{i,2}, \ldots, B_{i,k_i-1}, b_{i,k_i-1}, B_{i,k_i}$, and (ii) $v_i \in V(B_{i,1}) \setminus \{b_{i,1}\}$.

Gao and Richter used this structural result in order to inductively prove that every triconnected planar graph (in fact, every circuit graph) has a *closed 2-walk*, which is a walk on the graph starting and ending at the same vertex and passing through each vertex of the graph at least once and at most twice.

The same result is used by Leighton and Moitra to inductively prove that, for every circuit graph (G, C) (and hence every triconnected planar graph G), a Christmas cactus graph S spanning G exists. In fact, the outline of their algorithm for spanning G consists of the following steps: (i) use Gao and Richter's structural result to find chains of blocks $B_{i,1}, b_{i,1}, B_{i,2}, b_{i,2}, \ldots, B_{i,k_i-1}, b_{i,k_i-1}, B_{i,k_i}$ spanning all vertices of G not in C; (ii) inductively compute Christmas cactus graphs spanning each block $B_{i,j}$ (which is in turn a circuit graph); (iii) glue the Christmas cactus graphs spanning the blocks and C into a unique Christmas cactus graph spanning G.

Our spanning algorithm, as discussed above, finds the spanning graph of G without using Gao and Richter's result. Moreover, once one has a non-triangulated binary cactus spanning a triconnected planar graph G, it is easy to find a closed 2-walk that passes only through the edges of such a spanning

graph. Hence, our algorithm for spanning triconnected planar graphs also provides an alternative proof that every triconnected planar graph has a closed 2-walk.

It is worth observing that our algorithm for spanning a triconnected planar graph with a non-triangulated binary cactus works more generally for circuit graphs (as the Leighton and Moitra's algorithm). In fact, in our algorithm, the only graph which may contain separating pairs before Action 1 is actually a circuit graph, since all its separating pairs are incident to the outer face. A spanning cactus for such a graph can hence be found with the same algorithm described above.

6.6 Conclusions and Open Problems

In this chapter we have shown an algorithm for constructing greedy drawings of triangulations. The algorithm relies on two main results. The first one states that every triangulated binary cactus admits a greedy drawing. The second one states that, for every triangulation G, there exists a triangulated binary cactus S spanning G. Then, we have shown how to modify the algorithm provided for triangulations in order to deal with triconnected planar graphs thus proving a conjecture by Papadimitriou and Ratajczak [PR05], that was independently settled by Leighton and Moitra [LM08].

The main drawback of our algorithm (and of Leighton and Moitra's algorithm, as well) is that it uses real coordinates, hence it constructs drawings requiring exponential area once a finite resolution rule has been fixed. It would be interesting to understand whether such exponential area is necessary in some cases or there exists an algorithm to produce polynomial-area greedy drawings for triangulations and triconnected planar graphs. We deal with a related problem in the next Chapter of this thesis. In fact, we study the area requirements for greedy-drawable graphs in general and we show that there exist greedy-drawable trees requiring exponential area in any greedy drawing.

Although greedy drawings have been proved to be feasible for large classes of planar graphs (like triconnected planar graphs), a characterization of the graphs that admit a greedy drawing seems still to be an elusive goal.

Open Problem 6.1 *Characterize the class of (planar) graphs that admit a greedy drawing.*

A stronger version of the Papadimitriou and Ratajczak's conjecture [PR05] says that for every triconnected planar graph there exists a convex greedy

drawing. Although some partial positive results are known [Dha08, GS09a], the following problem is still open:

Open Problem 6.2 *Does a convex greedy drawing of every triconnected planar graph exist?*

Finally, most of the known algorithms for constructing greedy graph drawings rely on the knowledge of the entire graph topology. Designing distributed algorithms for computing greedy drawings or proving that such algorithms do not exist would be theoretically interesting and useful in practice for greedy routing.

Chapter 7

Succinct Representation of Greedy Drawings

In this chapter[1], we consider the problem of constructing greedy drawings in the plane with a small area. Such a problem is worth to study not only from the Graph Drawing perspective, but also from the greedy routing one, as in a greedy drawing with polynomial area the Cartesian coordinates of the vertices can be represented with few bits, which is a necessary condition for making the greedy routing useful in practice.

However, having polynomial area greedy drawings is not always possible, at least for trees. In fact, we prove that there exists an infinite class of caterpillars requiring exponential area in any greedy drawing.

Observe that the main theorem of this chapter is one of the few results (e.g., [DTT92]) showing that certain families of graph drawings require exponential area. Also, greedy drawings are a kind of proximity drawings [DLL95], a class of graph drawings, including Euclidean Minimum Spanning Trees [MS92, Kau08], for which very little is known about the area requirements [PV04].

7.1 Introduction

As discussed in the previous chapter, it is possible to construct greedy drawings of any given triconnected planar graph [LM08, AFG10]. However, the drawings constructed by the algorithms presented so far have the undesired property to

[1] Part of the contents of this chapter are a joint work with Giuseppe Di Battista and Fabrizio Frati, appeared in [ADF09].

be not *succinct*, i.e. they require $\Omega(n \log n)$ bits in the worst case for representing the vertex coordinates, as their construction requires exponential area. This makes them unsuitable for the motivating application of greedy routing, as such a number of bits is asymptotically equivalent to the number of bits needed to explicitly represent the information about how to reach each node of the network in a classical routing table.

Concerning the problem of representing vertex coordinates in a succinct way, Eppstein and Goodrich [EG08] proposed an elegant algorithm for greedy routing in the hyperbolic plane representing vertex coordinates with $O(\log n)$ bits. A similar result was obtained by Goodrich and Strash [GS09b] for greedy routing in the Euclidean plane, where the positions of the vertices are represented in an appropriate coordinate system which is strictly related to the way in which the drawing is constructed.

However, the perhaps most natural question of whether greedy drawings can be constructed in the plane using $O(\log n)$ bits for representing vertex Cartesian coordinates and using the Euclidean distance as a metric was, up to now, open. Observe that, when the Cartesian coordinates system and the Euclidean distance are used, this problem is equivalent to the one in which polynomial-area greedy drawings are requested.

In this chapter, we give a negative answer to the above question.

Theorem 7.1 *For infinitely many n, there exists a $(3n + 3)$-node greedy-drawable tree that requires b^n area in any greedy drawing in the plane, under any resolution rule, for some constant $b > 1$.*

We prove the theorem by showing that, in any greedy drawing of a suitably defined $(3n + 3)$-node greedy-drawable caterpillar T_n, the ratio between the length of the longest edge and the length of the shortest edge is exponential in n. Hence, once a resolution rule stating that the shortest edge has one unit length has been fixed, the length of the longest edge, and hence the area of the drawing, is exponential in n.

The chapter is organized as follows. In Sect. 7.2, we introduce some definitions and preliminaries; in Sect. 7.3 we prove that there exists an n-node tree T_n requiring exponential area in any greedy drawing; in Sect. 7.4 we show an algorithm for constructing a greedy drawing of T_n; finally, in Sect. 7.5 we conclude and present some open problems.

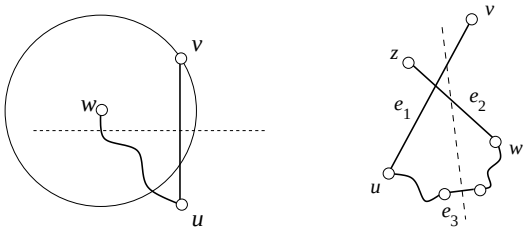


Figure 7.1: (a) If w lies in the half-plane delimited by the axis of (u, v) and containing v, then the drawing is not greedy. (b) A crossing in the drawing determines a violation of Lemma 7.3.

7.2 Definitions and Preliminaries

The *cell* of a node v in a drawing is the set of all the points in the plane that are closer to v than to any of its neighbors. The following lemma, due to Papadimitriou and Ratajczak [PR05], can be used to state greedy drawings in terms of proximity drawings.

Lemma 7.1 *[PR05] A drawing is greedy if and only if the cell of each node v contains no node other than v.*

We remark that the cell of a leaf node v is the half-plane containing v and delimited by the axis of the segment having v as an endpoint.

We now state some basic properties of the greedy drawings of trees.

Lemma 7.2 *Given a greedy drawing Γ of a tree T, any subtree of T is represented in Γ by a greedy drawing.*

Proof: Suppose, for a contradiction, that a subtree T' of T exists not represented in Γ by a greedy drawing. Then, there exist two nodes u and v such that the only path in T' from u to v is not distance-decreasing. However, such a path is also the only path from u to v in T, a contradiction. □

Lemma 7.3 *Given a greedy drawing Γ of a tree T and given any edge (u, v) of T, the subtree T' of T obtained by removing edge (u, v) from T that contains u (resp. v) completely lies in Γ in the half-plane containing u (resp. v) and delimited by the axis of (u, v).*

Proof: Refer to Fig. 7.1(a). Suppose, for a contradiction, that there exists a node w of T' that lies in Γ in the half-plane containing v (resp. u) and delimited by the axis of (u, v). Then, $d(v, w) < d(u, w)$ (resp. $d(u, w) < d(v, w)$). The only path from v to w (resp. from u to w) in T passes through u (resp. through v), hence it is not distance-decreasing, a contradiction. □

Lemma 7.4 *Any greedy drawing of a tree is planar.*

Proof: Refer to Fig. 7.1(b). Suppose, for a contradiction, that there exists a tree T admitting a non-planar greedy drawing Γ. Let $e_1 = (u, v)$ and $e_2 = (w, z)$ be two edges that cross in Γ. Edges e_1 and e_2 are not adjacent, otherwise they would overlap and Γ would not be greedy. Then, there exists an edge $e_3 \neq e_1, e_2$ in the only path connecting u to w. Lemma 7.3 implies that e_1 and e_2 lie in distinct half-planes delimited by the axis of e_3, hence they do not cross, a contradiction. □

Lemma 7.5 *In any greedy drawing of a tree T, the angle between two adjacent segments is strictly greater than* $60°$.

Proof: Consider any greedy drawing of T in which the angle between two adjacent segments $\overline{w_1w_2}$ and $\overline{w_2w_3}$ is no more than $60°$. Then, $|\overline{w_1w_3}| \leq |\overline{w_1w_2}|$ or $|\overline{w_1w_3}| \leq |\overline{w_2w_3}|$, say $|\overline{w_1w_3}| \leq |\overline{w_2w_3}|$. Since $d(w_1, w_3) \leq d(w_2, w_3)$, the unique path (w_1, w_2, w_3) from w_1 to w_3 in T is not distance-decreasing. □

In the following we define a family of trees with $3n + 3$ nodes, for every $n \geq 2$, that will be exploited in order to prove Theorem 7.1. Refer to Fig. 7.2.

Let T_n be a caterpillar with spine $(v_1, v_2, \ldots, v_n)$ such that v_1 has degree 5 and v_i has degree 4, for each $i = 2, 3, \ldots, n$. Let a_1, b_1, c_1, and d_1 be the leaves of T_n adjacent to v_1, let a_i and b_i be the leaves of T_n adjacent to v_i, for $i = 2, 3, \ldots, n-1$, and let a_n, b_n, and c_n be the leaves of T_n adjacent to v_n.

Distinct embeddings of T_n differ for the order of the edges incident to the spine nodes. More precisely, the clockwise order of the edges incident to each node v_i is one of the following: 1) (v_{i-1}, v_i), then a leaf edge, then (v_i, v_{i+1}), then a leaf edge: v_i is a *central node* (node v_n in Fig. 7.2.b); 2) (v_{i-1}, v_i), then two leaf edges, then (v_i, v_{i+1}): v_i is a *bottom node* (node v_2 in Fig. 7.2.b); or 3) (v_{i-1}, v_i), then (v_i, v_{i+1}), then two leaf edges: v_i is a *top node* (node v_3 in Fig. 7.2.b). Node v_1 is considered as a central node.

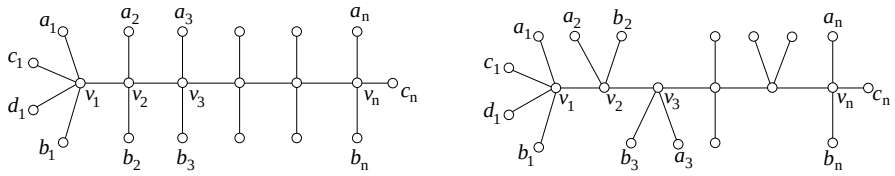


Figure 7.2: Two embeddings of caterpillar T_n. In (a) all the spine nodes are central nodes. In (b) node v_2 is a bottom node and node v_3 is a top node.

7.3 The Lower Bound

In this section we prove that any greedy drawing of T_n requires exponential area. The proof is based on the following intuitions: (i) For any central node v_i there exists a "small" convex region containing all the spine nodes v_j, with $j > i$, and their adjacent leaves (Lemma 7.6). (ii) Almost all the spine nodes are central nodes (Lemma 7.8). (iii) The slopes of edges (v_i, a_i), (v_i, v_{i+1}), and (v_i, b_i) incident to a central node v_i are in a certain range, which is more restricted for the edges incident to v_{i+1} than for those incident to v_i (Lemma 7.6). (iv) If the angle between (v_i, a_i) and (v_i, b_i) is too small, then v_j, a_j, and b_j, with $j \geq i+2$, can not be drawn (Lemma 7.10). (v) If both the angles between (v_i, a_i) and (v_i, b_i) and between (v_{i+1}, a_{i+1}) and (v_{i+1}, b_{i+1}) are large enough, then the ratio between the length of the edges incident to v_i and the length of the edges incident to v_{i+1} is constant (Lemma 7.9).

First, we discuss some properties of the slopes of the edges in the drawing. Second, we argue about the exponential decrease of the edge lengths.

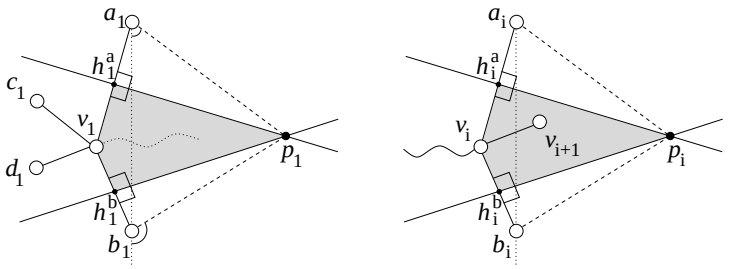
Slopes

Consider any drawing of v_1 and of its adjacent leaves; rename such leaves so that the counter-clockwise order of the vertices around v_1 is a_1, c_1, d_1, b_1, v_2.

In the following, when we refer to an angle $\widehat{v_1 v_2 v_3}$, we mean the angle that brings the half-line from v_2 through v_1 to coincide with the half-line from v_2 through v_3 by a counter-clockwise rotation.

Property 7.1 $\widehat{b_1 v_1 a_1} < 180°$.

Proof: By Lemma 7.5, $\widehat{a_1 v_1 c_1} > 60°$, $\widehat{c_1 v_1 d_1} > 60°$, and $\widehat{d_1 v_1 b_1} > 60°$. □



(a) (b)

Figure 7.3: (a) Region R_1 contains the drawing of $T_n \setminus \{a_1, b_1, c_1, d_1, v_1\}$. The slopes of $\overline{a_1p_1}$ and $\overline{b_1p_1}$ are shown. (b) Region R_i contains the drawing of path $(v_{i+1}, v_{i+2}, \ldots, v_n)$ and of its adjacent leaves.

We argue that, for any central node v_i, there exists a "small" convex region containing all the spine nodes v_j, with $j > i$, and their adjacent leaves.

Let v_i be a central node and suppose that $\widehat{b_iv_ia_i} < 180°$. Denote by R_i the convex region delimited by $\overline{v_ia_i}$, by $\overline{v_ib_i}$, and by the axes of such segments (see Fig. 7.3(b)). Denote by p_i the intersection between the axes of $\overline{v_ia_i}$ and $\overline{v_ib_i}$, and by h_i^a (h_i^b) the midpoint of $\overline{v_ia_i}$ (resp. $\overline{v_ib_i}$).

Assume that $x(a_i) = x(b_i)$, $x(v_i) < x(a_i)$, and $y(a_i) > y(b_i)$. Such a setting can be achieved w.l.o.g. up to a rotation/mirroring of the drawing and a renaming of the leaves. In the following, whenever a central node v_i is considered, the drawing is rotated/mirrored and the leaves adjacent to v_i are renamed so that $x(a_i) = x(b_i)$, $x(v_i) < x(a_i)$, and $y(a_i) > y(b_i)$.

Let $slope(u, v)$ be the angle bringing the half-line from u directed downward to coincide with the half-line from u through v by a counter-clockwise rotation (see Fig. 7.3(a)). Further, let $slope_\perp(u, v)$ be equal to $slope(u, v) - 90°$. We observe the following:

Property 7.2 *$slope(v_i, b_i) < slope_\perp(b_i, p_i) < slope_\perp(p_i, a_i) < slope(v_i, a_i)$.*

Proof: Inequality $slope(v_i, b_i) < slope_\perp(b_i, p_i)$ (and analogously inequality $slope_\perp(p_i, a_i) < slope(v_i, a_i)$) holds since $slope(h_i^b, p_i) < slope(b_i, p_i)$. Inequality $slope_\perp(b_i, p_i) < slope_\perp(p_i, a_i)$ holds by assumption. □

Lemma 7.6 *Suppose that v_i is a central node. Then, the following hold: (i) $\widehat{b_iv_ia_i} < 180°$; (ii) the drawing of path $(v_{i+1}, v_{i+2}, \ldots, v_n)$ and of its adjacent*

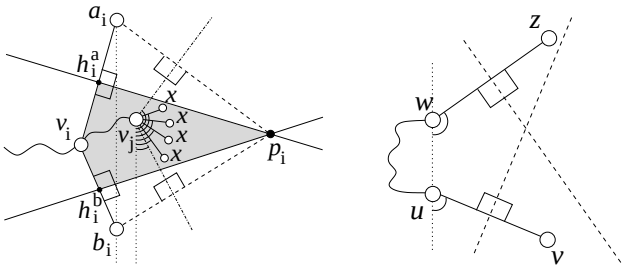


Figure 7.4: (a) Possible slopes for an edge (v_j, x). (b) Illustration for the proof of Lemma 7.7.

leaves lies in R_i; and (iii) any edge (v_j, x), where $x \in \{a_j, b_j, v_{j+1}\}$ with $j > i$, is such that $slope_\perp(b_i, p_i) < slope(v_j, x) < slope_\perp(p_i, a_i)$. See Fig. 7.4(a).

Proof: When $i = 1$, Property 7.1 ensures part (i). Further, Lemma 7.1 ensures part (ii), that is, the drawing of $T_n \setminus \{a_1, b_1, c_1, d_1, v_1\}$ lies in R_1 (see Fig. 7.3(a)). In order to prove that part (iii) of the lemma holds when $i = 1$, suppose, for a contradiction, that an edge (v_j, x) exists, where $x \in \{a_j, b_j, v_{j+1}\}$ with $j > 1$, such that $slope_\perp(b_1, p_1) < slope(v_j, x) < slope_\perp(p_1, a_1)$ does not hold. Then, it is easy to see that the half-plane delimited by the axis of $\overline{v_j x}$ and containing x also contains at least one out of a_1, v_1, and b_1, thus providing a contradiction to the greediness of the drawing, by Lemma 7.3.

By induction, suppose that part (i), part (ii), and part (iii) of the lemma hold for some i. Let k be the smallest index greater than i such that v_k is a central node. Then, by part (iii) of the inductive hypothesis and by Property 7.2, $slope(v_i, b_i) < slope_\perp(b_i, p_i) < slope(v_k, b_k) < slope(v_k, a_k) < slope_\perp(p_i, a_i) < slope(v_i, a_i)$ holds, which implies $\widehat{b_k v_k a_k} < \widehat{b_i v_i a_i} < 180°$, and part (i) of the lemma follows for k.

By Lemma 7.4, the drawing is planar; by Lemma 7.1, the cells of a_k and b_k do not contain any node other than a_k and b_k, respectively. Hence, if a node u is in R_k, then no node of any subtree of T_n containing u and not containing v_k lies outside R_k. Thus, v_{k-1} does not lie in R_k (since a subtree of T_n exists containing v_{k-1}, v_i, and not containing v_k); since v_k is a central node, then v_{k+1} lies on the opposite side of v_{k-1} with respect to the path composed of edges (v_k, a_k) and (v_k, b_k). Hence, v_{k+1} (and path $(v_{k+1}, v_{k+2}, \ldots, v_n)$ together with its adjacent leaves) lies inside R_k, and part (ii) of the lemma follows for

k.

Once parts (i) and (ii) of the lemma have been proved for k, part (iii) of the lemma can also be proved for k analogously as in the base case. Namely, if $slope_{\perp}(b_k, p_k) < slope(v_j, x) < slope_{\perp}(p_k, a_k)$ does not hold, for some edge (v_j, x) with $j > k$, then the half-plane delimited by the axis of $\overline{v_j x}$ and containing x also contains at least one out of a_k, v_k, and b_k, thus implying that the drawing is not greedy, by Lemma 7.3. □

Consider two edges (u, v) and (w, z) such that the path from u to w does not contain v and z. Suppose, w.l.o.g. up to a rotation/mirroring of the drawing, that v and z lie in the same half-plane delimited by the line through u and w, and that $x(u) = x(w)$, $y(u) < y(w)$, and $0° < slope(u, v), slope(w, z) < 180°$.

Lemma 7.7 *$slope(u, v) < slope(w, z)$.*

Proof: Refer to Fig. 7.4(b). Suppose, for a contradiction, that $slope(u, v) \geq slope(w, z)$. Then, either v lies in the half-plane delimited by the axis of (w, z) and containing z, or z lies in the half-plane delimited by the axis of (u, v) and containing v. Hence, by Lemma 7.2, the drawing is not greedy. □

Exponential Decreasing Edge Lengths

Now we are ready to go in the mainstream of the proof that any greedy drawing of T_n requires exponential area. Such a proof is in fact based on the following three lemmata. The first one states that a linear number of spine nodes are central nodes, in any greedy drawing of T_n.

Lemma 7.8 *Suppose that v_i is a central node, for some $i \leq n-3$. Then, v_{i+1} is a central node.*

Proof: Refer to Fig. 7.5. Suppose, for a contradiction, that v_{i+1} is not a central node. Suppose that v_{i+1} is a top node, the case in which it is a bottom node being analogous. Rename the leaves adjacent to v_{i+1} in such a way that the counter-clockwise order of the neighbors of v_{i+1} is v_i, b_{i+1}, a_{i+1}, and v_{i+2}. By Lemma 7.6 part (i), $\widehat{b_i v_i a_i} < 180°$. By Lemma 7.6 part (iii), by Property 7.2, and by the assumption that v_{i+1} is a top node, $slope(v_i, b_i) < slope(v_{i+1}, b_{i+1}) < slope(v_{i+1}, a_{i+1}) < slope(v_{i+1}, v_{i+2}) < slope(v_i, a_i)$. By Lemma 7.5, $\widehat{b_{i+1} v_{i+1} a_{i+1}} > 60°$. It follows that $\widehat{a_{i+1} v_{i+1} v_{i+2}} < 120°$.

Suppose that v_{i+2} is a central node (a top node; a bottom node). Rename the leaves adjacent to v_{i+2} in such a way that the counter-clockwise order of the neighbors of v_{i+2} is v_{i+1}, b_{i+2}, v_{i+3}, and a_{i+2} (resp. v_{i+1}, b_{i+2},

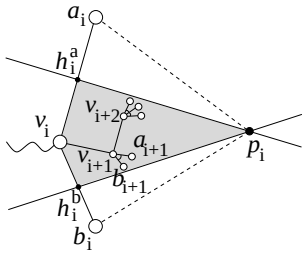


Figure 7.5: Illustration for the proof of Lemma 7.8.

a_{i+2}, and v_{i+3}; v_{i+1}, v_{i+3}, b_{i+2}, and a_{i+2}). Notice that node v_{i+3} exists since $i \leq n-3$. By Lemma 7.7, $slope(v_{i+2}, b_{i+2}) > slope(v_{i+1}, a_{i+1})$ (resp. $slope(v_{i+2}, b_{i+2}) > slope(v_{i+1}, a_{i+1})$; $slope(v_{i+2}, v_{i+3}) > slope(v_{i+1}, a_{i+1})$). Further, by Lemma 7.6 part (iii), it holds $slope(v_{i+2}, a_{i+2}) < slope(v_i, a_i)$ (resp. $slope(v_{i+2}, v_{i+3}) < slope(v_i, a_i)$; $slope(v_{i+2}, a_{i+2}) < slope(v_i, a_i)$). It follows that $b_{i+2}\widehat{v_{i+2}}a_{i+2} < 120°$ (resp. $b_{i+2}\widehat{v_{i+2}}v_{i+3} < 120°$; $v_{i+3}\widehat{v_{i+2}}a_{i+2} < 120°$), hence at least one of $b_{i+2}\widehat{v_{i+2}}v_{i+3}$ and $v_{i+3}\widehat{v_{i+2}}a_{i+2}$ (resp. of $b_{i+2}\widehat{v_{i+2}}a_{i+2}$ and $a_{i+2}\widehat{v_{i+2}}v_{i+3}$; of $v_{i+3}\widehat{v_{i+2}}b_{i+2}$ and $b_{i+2}\widehat{v_{i+2}}a_{i+2}$) is less than 60°. By Lemma 7.5, the drawing is not greedy. □

The next lemma shows that, if the angles $\widehat{b_i v_i a_i}$ incident to each central node v_i are large enough, then the sum of the lengths of $\overline{v_i a_i}$ and $\overline{v_i b_i}$ decreases exponentially in the number of considered central nodes.

Lemma 7.9 *Let v_i be a central node, with $i \leq n-3$. Suppose that both the angles $\widehat{b_i v_i a_i}$ and $b_{i+1}\widehat{v_{i+1}}a_{i+1}$ are greater than 150°. Then, the following inequality holds: $|\overline{v_{i+1}a_{i+1}}| + |\overline{v_{i+1}b_{i+1}}| \leq (|\overline{v_i a_i}| + |\overline{v_i b_i}|)/\sqrt{3}$.*

Proof: Refer to Fig. 7.6(a). By Lemma 7.8, v_{i+1} is a central node. Denote by $l(v_{i+1})$ the vertical line through v_{i+1} and denote by $l(h_i^a)$ and $l(h_i^b)$ the horizontal lines through h_i^a and h_i^b, respectively.

By Lemma 7.6 part (iii), $slope_\perp(b_i, p_i) < slope(v_{i+1}, b_{i+1}) < slope(v_{i+1}, a_{i+1}) < slope_\perp(p_i, a_i)$. Hence, by Property 7.2, we have $slope(v_i, b_i) < slope(v_{i+1}, b_{i+1}) < slope(v_{i+1}, a_{i+1}) < slope(v_i, a_i)$. It follows that both a_{i+1} and b_{i+1} lie in the half-plane delimited by $l(v_{i+1})$ and not containing v_i. Denote by d_{i+1}^a (d_{i+1}^b) the intersection point between $l(v_{i+1})$ and $l(h_i^a)$ (resp. and $l(h_i^b)$). Observe that $|\overline{d_{i+1}^b d_{i+1}^a}| < (|\overline{v_i b_i}| + |\overline{v_i a_i}|)/2$. Denote by f_{i+1}^a (by f_{i+1}^b) the

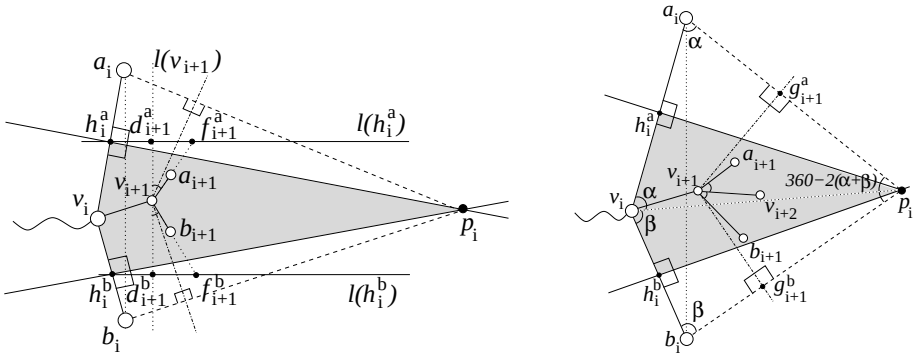


Figure 7.6: Illustrations for the proofs of Lemma 7.9 (a) and Lemma 7.10 (b).

intersection point between $l(h_i^a)$ and the line through v_{i+1} and a_{i+1} (resp. between $l(h_i^b)$ and the line through v_{i+1} and b_{i+1}). Clearly, $|\overline{v_{i+1}a_{i+1}}| < |\overline{v_{i+1}f_{i+1}^a}|$ and $|\overline{v_{i+1}b_{i+1}}| < |\overline{v_{i+1}f_{i+1}^b}|$. Angles $\widehat{d_{i+1}^b v_{i+1} f_{i+1}^b}$ and $\widehat{f_{i+1}^a v_{i+1} d_{i+1}^a}$ are each less than 30°, namely such angles sum up to an angle which is 180° minus $\widehat{f_{i+1}^b v_{i+1} f_{i+1}^a}$, which by hypothesis is greater than 150°. Hence, $|\overline{v_{i+1}a_{i+1}}| < |\overline{v_{i+1}f_{i+1}^a}| < |\overline{v_{i+1}d_{i+1}^a}|/\cos(30)$ and $|\overline{v_{i+1}b_{i+1}}| < |\overline{v_{i+1}f_{i+1}^b}| < |\overline{v_{i+1}d_{i+1}^b}|/\cos(30)$. It follows that $|\overline{v_{i+1}a_{i+1}}| + |\overline{v_{i+1}b_{i+1}}| < (|\overline{v_{i+1}d_{i+1}^a}| + |\overline{v_{i+1}d_{i+1}^b}|)/\cos(30) < 2(|\overline{v_ib_i}| + |\overline{v_ia_i}|)/2\sqrt{3}$, thus proving the lemma. □

The next lemma shows that having large angles incident to central nodes is unavoidable for almost all central nodes.

Lemma 7.10 *No central node v_i, with $i \leq n-3$, is incident to an angle $\widehat{b_iv_ia_i}$ that is less than or equal to* 150°.

Proof: Refer to Fig. 7.6(b). Suppose, for a contradiction, that there exists a central node v_i, with $i \leq n-3$, that is incident to an angle $\widehat{b_iv_ia_i} \leq 150°$. Denote by α and β the angles $\widehat{p_iv_ia_i}$ and $\widehat{b_iv_ip_i}$, respectively. Since triangles (v_i, p_i, h_i^a) and (a_i, p_i, h_i^a) are congruent, $\widehat{v_ia_ip_i} = \alpha$. Analogously, $\widehat{v_ib_ip_i} = \beta$. Summing up the angles of quadrilateral (v_i, a_i, p_i, b_i), we get $\widehat{a_ip_ib_i} = 360° - 2(\alpha+\beta)$.

By Lemma 7.8, v_{i+1} is a central node. Consider the line through v_{i+1} orthogonal to $\overline{a_ip_i}$ and denote by g^a_{i+1} the intersection point between such a line and $\overline{a_ip_i}$. Further, consider the line through v_{i+1} orthogonal to $\overline{b_ip_i}$ and denote by g^b_{i+1} the intersection point between such a line and $\overline{b_ip_i}$. By Lemma 7.6 part (iii), $slope_\perp(b_i, p_i) < slope(v_{i+1}, b_{i+1}) < slope(v_{i+1}, a_{i+1}) < slope_\perp(p_i, a_i)$. Hence, $\widehat{b_{i+1}v_{i+1}a_{i+1}} < \widehat{g^b_{i+1}v_{i+1}g^a_{i+1}}$. Further, $\widehat{g^b_{i+1}v_{i+1}g^a_{i+1}} = 2\alpha + 2\beta - 180°$, as can be derived by considering quadrilateral $(g^b_{i+1}, v_{i+1}, g^a_{i+1}, p_i)$. Since, by hypothesis, $\alpha + \beta \leq 150°$, we have $\widehat{b_{i+1}v_{i+1}a_{i+1}} < \widehat{g^b_{i+1}v_{i+1}g^a_{i+1}} = 2\alpha + 2\beta - 180° \leq 120°$. However, since v_{i+1} is a central node, edge $(v_{i+1}v_{i+2})$, that exists since $i \leq n-3$, cuts angle $\widehat{b_{i+1}v_{i+1}a_{i+1}}$. It follows that at least one of angles $\widehat{b_{i+1}v_{i+1}v_{i+2}}$ and $\widehat{v_{i+2}v_{i+1}a_{i+1}}$ is less than $60°$. By Lemma 7.5, the drawing is not greedy. □

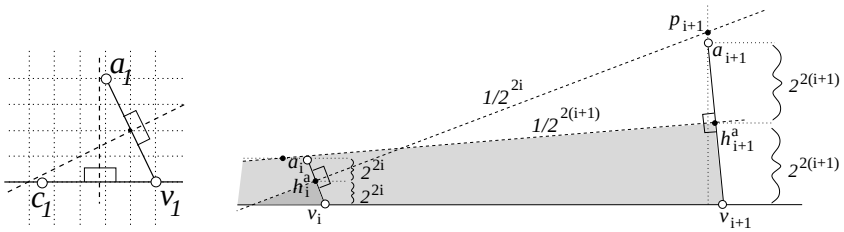
The previous lemmata immediately imply an exponential lower bound between the ratio of the lengths of the longest and the shortest edge of the drawing. Namely, node v_1 is a central node. By Lemma 7.8, v_i is a central node, for $i = 2, \ldots, n-3$. By Lemma 7.10, angle $\widehat{b_iv_ia_i} > 150°$, for each $i \leq n-3$. Hence, by Lemma 7.9, $|\overline{v_{i+1}a_{i+1}}| + |\overline{v_{i+1}b_{i+1}}| \leq (|\overline{v_ia_i}| + |\overline{v_ib_i}|)/\sqrt{3}$, for each $i \leq n-4$; it follows that $|\overline{v_{n-3}a_{n-3}}| + |\overline{v_{n-3}b_{n-3}}| \leq (|\overline{v_1a_1}| + |\overline{v_1b_1}|)/(\sqrt{3})^{n-4}$. Since one out of $\overline{v_1a_1}$ and $\overline{v_1b_1}$, say $\overline{v_1a_1}$, has length at least half of $|\overline{v_1a_1}| + |\overline{v_1b_1}|$, and since one out of $\overline{v_{n-3}a_{n-3}}$ and $v_{n-3}b_{n-3}$, say $\overline{v_{n-3}a_{n-3}}$, has length at most half of $|\overline{v_{n-3}a_{n-3}}| + |\overline{v_{n-3}b_{n-3}}|$, then $|\overline{v_1a_1}|/|\overline{v_{n-3}a_{n-3}}| \geq \frac{1}{9}(\sqrt{3})^n$, thus implying the claimed lower bound.

7.4 Drawability of T_n

In Sect. 7.3 we have shown that any greedy drawing of T_n requires exponential area. Since in [PR05, LM08] it has been shown that there exist trees that do not admit any greedy drawing, one might ask whether the lower bound refers to a greedy-drawable tree or not. Of course, if T_n were not drawable, then the lower bound would not make sense. In this section we show that T_n admits a greedy drawing by providing a drawing algorithm, using a supporting exponential-size grid.

Since the algorithm draws the spine nodes in the order they appear on the spine with the degree-5 node as the last node, we revert the indices of the nodes with respect to Sects. 7.2 and 7.3, that is, node v_i of T_n is now node v_{n-i+1}.

The algorithm constructs a drawing of T_n in which all the spine nodes v_i are central nodes lying on the horizontal line $y = 0$. Since each leaf node a_i is



(a) (b)

Figure 7.7: Illustrations for the algorithm to construct a greedy drawing of T_n. (a) Base case. (b) Inductive case.

drawn above line $y = 0$ and b_i is placed on the symmetrical point of a_i with respect to such a line, we only describe, for each $i = 1, \ldots, n$, how to draw v_i and a_i.

In order to deal with drawings that lie on a grid, in this section we denote by Δ_y/Δ_x the *slope* of a line (of a segment), meaning that whenever there is a horizontal distance Δ_x between two nodes of such a line (of such a segment), then their vertical distance is Δ_y.

The drawing is constructed by means of an inductive algorithm. In the base case, place v_1 at $(0, 0)$, h_1^a at $(-1, 2)$, a_1 at $(-2, 4)$, and c_1 at $(-9/2, 0)$ (see Fig. 7.7(a)). At step i of the algorithm suppose, by inductive hypothesis, that: (i) The drawing of path $(v_1, v_2, \ldots, v_i)$ with its leaf nodes $a_1, a_2, \ldots, a_i$ is greedy, and (ii) $y(v_i) = 0$, $y(h_i^a) = 2^{2i}$, $y(a_i) = 2^{2i+1}$, and $x(v_i) - x(h_i^a) = x(h_i^a) - x(a_i) = 1$.

From the above inductive hypothesis it follows that the slope of segment $\overline{v_i a_i}$ is $-2^{2i}/1$ and the slope of its axis is $1/2^{2i}$. We show step $i + 1$ of the algorithm.

Place v_{i+1} at point $(x(v_i) + 2^{4i+3} - 2, 0)$, h_{i+1}^a at point $(x(v_i) + 2^{4i+3} - 3, 2^{2i+2})$, and a_{i+1} at point $(x(v_i) + 2^{4i+3} - 4, 2^{2i+3})$ (see Fig. 7.7(b)). Such placements guarantee that part (ii) of the hypothesis is verified. The slope of segment $\overline{v_{i+1}a_{i+1}}$ is $-2^{2(i+1)}/1$. Hence, the slope of its axis is $1/2^{2(i+1)}$. Such an axis passes through point $q_i \equiv (x(v_i) - 3, 2^{2i+1})$. Since $0 < 1/2^{2(i+1)} < 1/2^{2i}$, it follows that path $(v_1, v_2, \ldots, v_i)$ with nodes $a_1, a_2, \ldots, a_i$ lie below the axis of $\overline{v_{i+1}a_{i+1}}$. Finally, the axis of $\overline{v_i a_i}$ passes through point $p_{i+1} \equiv (x(v_i) + 2^{4i+3} - 4, 2^{2i} + 2^{2i+3} - 3/2^{2i})$. Thus, $y(p_{i+1}) > y(a_{i+1})$, since $2^{2i} + 2^{2i+3} - 3/2^{2i} > 2^{2i+3}$ as long as $2^{4i} > 3$, which holds for each $i \geq 1$. This implies that part (i) of the hypothesis is verified.

When the algorithm has drawn v_n and a_n (and symmetrically b_n), c_n and d_n still have to be drawn. However, this can be easily done by assigning to segments $\overline{v_n c_n}$ and $\overline{v_n d_n}$ the same length as segment $\overline{v_n a_n}$ and by placing them in such a way that the angle $\widehat{b_n v_n a_n}$, which is strictly greater than 180°, is split into three angles strictly greater than 60°.

We remark that c_n and d_n are not placed at points with rational coordinates. However, they still obey to any resolution rule, namely their distance from any node or edge of the drawing is exponential with respect to the grid unit. Placing such nodes at grid points is possible after a scaling of the whole drawing and some non-trivial calculations. However, we preferred not to deal with such an issue since we just needed to prove that a greedy drawing of T_n exists.

7.5 Conclusions

We have shown that constructing succinct greedy drawings in the plane, when the Euclidean distance is adopted, may be unfeasible even for simple classes of trees. In fact, we proved that there exist caterpillars requiring exponential area in any greedy drawing, under any finite resolution rule. The proof uses a mixed geometric-topological technique that allows us to analyze the combinatorial space of the possible embeddings and to identify nice invariants of the slopes of the edges in any greedy drawing of such caterpillars.

Many problems remain open in this research field related to the area of greedy drawings. The most natural is summarized in the following question:

Open Problem 7.1 *What are the area requirements of greedy drawings of triangulations and triconnected planar graphs?*

Another intriguing question arises from the discussion presented in this Chapter. We have shown, in Lemma 7.4, that every greedy drawing of a tree is planar. It would be interesting to understand whether trees are the only class of planar graphs with such a property.

Open Problem 7.2 *Characterize the class of planar graphs such that every greedy drawing is planar.*

Part IV

Simultaneous Embedding of Planar Graphs

Chapter 8

Geometric Simultaneous Embedding of a Tree and a Path

In this chapter[1] we deal with the problem of simultaneously drawing two planar graphs on the same set of points in such a way that each of the drawings is planar. In particular, we consider the most natural version of the problem, that is, the one in which both the graphs have to be drawn with straight-line edges, that is called *geometric simultaneous embedding*.

Many results are known on this field, concerning the existence of such embeddings for some classes of graphs and the computational complexity of the corresponding decision problem. While it is known that two caterpillars always admit a geometric simultaneous embedding and that two trees not always admit one, the question about a tree and a path is still open and is often regarded as the most prominent open problem in this area.

We answer this question in the negative by providing a counterexample. Additionally, since the counterexample uses disjoint edge sets for the two graphs, we also negatively answer another open question, that is, whether it is possible to simultaneously embed two edge-disjoint trees.

As a final result of this chapter, we study the same problem when some constraints on the tree are imposed. Namely, we show that a tree of depth 2 and a path always admit a geometric simultaneous embedding. In fact, such a strong constraint is not so far from closing the gap with the instances not

[1] Parts of the contents of this chapter are a joint work with Markus Geyer, Michael Kaufmann, and Daniel Neuwirth, appeared in [AGKN10] and submitted to international conference.

admitting any solution, as the tree used in our counterexample has depth 4.

8.1 Introduction

Embedding planar graphs is a well-established field in graph theory and algorithms with a great variety of applications. Recently, motivated by the need of concurrently represent several different relationships among the same set of elements, a major focus in the research lies on *simultaneous graph embedding*. In this setting, given a set of graphs with the same vertex-set, the goal is to find a set of points in the plane and a mapping between these points and the vertices of the graphs such that placing each vertex on the point it is mapped to yields a planar drawing for each of the graphs, if they are displayed separately. Problems of this kind frequently arise when dealing with the visualization of evolving networks and with the visualization of huge and complex relationships, as in the case of the graph of the Web.

Among the many variants of this problem, the most important and natural one is the *geometric simultaneous embedding*. Given two graphs $G_1 = (V, E')$ and $G_2 = (V, E'')$, the task is to find a set of points P and a bijection $M : P \to V$ that induce planar straight-line drawings for both G_1 and G_2.

In the seminal paper on this topic [BCD+07], Brass *et al.* proved that geometric simultaneous embeddings of pairs of paths, pairs of cycles, and pairs of caterpillars always exist. On the other hand, many negative results have been shown. Brass *et al.* [BCD+07] presented a pair of outerplanar graphs not admitting any simultaneous embedding and provided negative results for three paths, as well. Erten and Kobourov [EK04] found a planar graph and a path not allowing any simultaneous embedding. Geyer *et al.* [GKV09] proved that there exist two trees that do not admit any geometric simultaneous embedding. However, the two trees used in the counterexample have common edges, and so the problem is still open for edge-disjoint trees. Finally, Cabello *et al.* [CvKL+09] showed a planar graph and a matching that do not admit any geometric simultaneous embedding and presented algorithms to obtain a geometric simultaneous embedding of a matching and a wheel, an outerpath, or a tree.

The most important open problem in this area is the question whether a tree and a path always admit a geometric simultaneous embedding or not. In this chapter we answer this question in the negative.

Many variants of the problem, where some constraints are relaxed, have been studied. If the edges do not need to be straight-line segments, any number

of planar graphs admit a simultaneous embedding, since any planar graph can be planarly embedded on any given set of points in the plane [Hal91, PW01]. However, the same result does not hold if the edges that are shared by two graphs have to be represented by the same Jordan curve. In this setting the problem is called *simultaneous embedding with fixed edges* [Fra06, GJP+06, FJKS08].

The research on this problem opened a new exciting field of problems and techniques, like ULP trees and graphs [EBFK09, FK07a, FK07b], colored simultaneous embedding [BEF+07], near-simultaneous embedding [FKK07], and matched drawings [DDv+07], deeply related to the general fundamental question of point-set embeddability.

In this chapter we answer the question about the geometric simultaneous embedding of a tree and a path in the negative by providing a counterexample. Moreover, since the tree and the path used in our counterexample do not share any edge, we also negatively answer the question on two edge-disjoint trees.

The main idea behind our counterexample is to use the path to enforce a part of the tree to be in a certain configuration which cannot be drawn planar. Namely, we make use of level nonplanar trees [EBFK09, FK07b], that is, trees not admitting any planar embedding if their vertices have to be placed inside certain regions according to a particular leveling. The tree of the counterexample contains many copies of such trees, while the path is used to create the regions. To prove that at least one copy has to be in the particular leveling that determines a crossing, we need a quite huge number of vertices. However, such a huge number is often needed just to ensure the existence of particular structures playing a role in our proof. A much smaller counterexample could likely be constructed with the same techniques, but we decided to prefer the simplicity of the arguments rather than the search for the minimum size.

The chapter is organized as follows. In Sect. 8.2 we give preliminary definitions and we introduce the concept of level nonplanar trees. In Sect. 8.3 we describe the tree $\mathcal{T}$ and the path $\mathcal{P}$ used in the counterexample. In Sect. 8.4 we give an overview of the proof that $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding, while in Sect. 8.5 we give the details of such a proof. In Sect. 8.6 we present an algorithm for the simultaneous embedding of a tree of depth 2 and a path, and in Sect. 8.7 we make some final remarks.

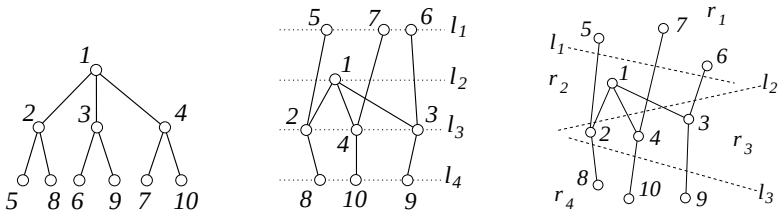


Figure 8.1: (a) A tree T_u. (b) A level nonplanar tree T whose underlying tree is T_u. (c) A region-level nonplanar tree T whose underlying tree is T_u.

8.2 Preliminaries

A (undirected) *k-level tree* $T = (V, E, \phi)$ on n vertices is a tree $T' = (V, E)$, called the *underlying tree* of T, together with a leveling of its vertices given by a function $\phi : V \mapsto \{1, \ldots, k\}$, such that for every edge $(u, v) \in E$, it holds $\phi(u) \neq \phi(v)$ (See [EBFK09, FK07b]). A drawing of $T = (V, E, \phi)$ is a *level drawing* if each vertex $v \in V$ such that $\phi(v) = i$ is placed on a horizontal line $l_i = \{(x, i) \mid x \in \mathbb{R}\}$. A level drawing of T is *planar* if no two edges intersect except, possibly, at common end-points. A tree $T = (V, E, \phi)$ is *level nonplanar* if it does not admit any planar level drawing.

We extend this concept to the one of *region-level drawing* by enforcing the vertices of each level to lie inside a certain region rather than on a horizontal line. Let $l_1, \ldots, l_k$ be k pairwise non-crossing straight-line segments and let $r_1, \ldots, r_{k+1}$ be the regions of the plane such that any straight-line segment connecting a point in r_i and a point in r_h, with $1 \leq i < h \leq k + 1$, cuts all and only the segments $l_i, l_{i+1}, \ldots, l_{h-1}$, in this order. A drawing of a k-level tree $T = (V, E, \phi)$ is called *region-level drawing* if each vertex $v \in V$ such that $\phi(v) = i$ is placed inside region r_i. A region-level drawing of T is *planar* if no two edges intersect except, possibly, at common end-points. A tree $T = (V, E, \phi)$ is *region-level nonplanar* if it does not admit any planar region-level drawing.

The 4-level tree T whose underlying tree is shown in Fig. 8.1(a) has been shown to be level nonplanar [FK07b] (see Fig. 8.1(b)). In the next lemma we show that T is also region-level nonplanar (see Fig. 8.1(c)).

Lemma 8.1 *The 4-level tree T whose underlying tree is shown in Fig. 8.1(a) is region-level nonplanar.*

Proof: Refer to Fig. 8.1(c). First observe that, in any possible region-level planar drawing of T, there exists a polygon Q_2 inside region r_2 delimited by paths $p_1 = v_5, v_2, v_8$ and $p_2 = v_6, v_3, v_9$, and by segments l_1 and l_2, and a polygon Q_3 inside region r_3 delimited by paths $p_1 = v_5, v_2, v_8$ and $p_2 = v_6, v_3, v_9$, and by segments l_2 and l_3. We have that v_1 is inside Q_2, as otherwise one of edges (v_1, v_2) or (v_1, v_3) would cross one of p_1 or p_2. Hence, vertex v_4 has to be inside Q_3, as otherwise edge (v_1, v_4) would cross one of p_1 or p_2. However, in this case, there is no placement for vertices v_7 and v_{10} that avoids a crossing between one of edges (v_4, v_7) or (v_4, v_{10}) and one of the already drawn edges. □

Lemma 8.1 will be vital for proving that there exist a tree $\mathcal{T}$ and a path $\mathcal{P}$ not admitting any geometric simultaneous embedding. In fact, $\mathcal{T}$ contains many copies of the underlying tree of T, while $\mathcal{P}$ connects vertices of $\mathcal{T}$ in such a way to create the regions satisfying the above conditions and to enforce at least one of such copies to lie inside these regions according to the leveling making it nonplanar.

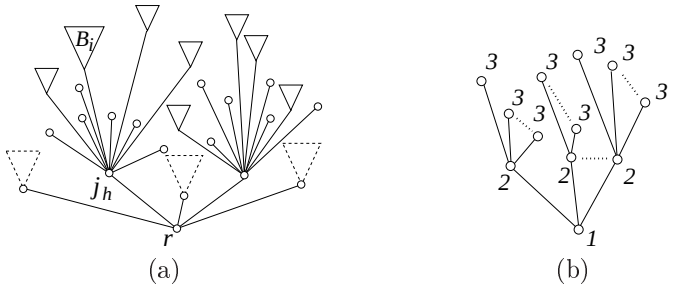
8.3 The Counterexample

In this section we describe a tree $\mathcal{T}$ and a path $\mathcal{P}$ not admitting any geometric simultaneous embedding.

Tree $\mathcal{T}$

The tree $\mathcal{T}$ contains a root r and q vertices $j_1, \ldots, j_q$ at distance 1 from r, called *joints*. Each joint j_h, with $h = 1, \ldots, q$, is connected to x copies $B_1, \ldots, B_x$ of a subtree, called *branches*, and to $l := (s-1)^4 \cdot 3^2 \cdot x$ vertices of degree 1, called *stabilizers*. See Fig. 8.2(a). Each branch B_i consists of a root r_i, $(s-1) \cdot 3$ vertices of degree $(s-1)$ adjacent to r_i, and $(s-2) \cdot (s-1) \cdot 3$ leaves at distance 2 from r_i. Vertices belonging to a branch B_i are called *B-vertices* and denoted by 1-, 2-, or 3-vertices, according to their distance from their joint. Fig. 8.2(b) displays 1-, 2-, and 3-vertices of a branch B_i.

Because of the huge number of vertices, in the rest of the chapter, for the sake of readability, we use variables n, s, and x as parameters describing the size of certain configurations. Such parameters will be given a value when the technical details of the arguments are described. At this stage we just claim



(a) (b)

Figure 8.2: (a) A schematization of the complete tree $\mathcal{T}$. Joints and stabilizers are small circles, branches are solid triangles, while complete subtrees connected to a joint are dashed triangles. (b) A schematization of a branch B_i.

that a total number $n \leq \binom{2^7 \cdot 3 \cdot x+2}{3}$ of vertices (see Lemmata 8.5 and 8.4) suffices for the counterexample.

As a first observation we note that, despite the oversized number of vertices, tree $\mathcal{T}$ has limited *depth*, that is, every vertex is at distance from the root at most 4. This leads to the following property.

Property 8.1 *Any path of tree edges starting at the root has at most* 3 *bends.*

Path $\mathcal{P}$

Path $\mathcal{P}$ is given by describing some basic and recurring subpaths on the vertices of $\mathcal{T}$ and how such subpaths are connected to each other. The idea is to partition the set of branches B_i adjacent to each joint j_h into subsets of s branches each and to connect their vertices with path edges, according to some features of the tree structure, so defining the first building block, called *cell*. Then, cells belonging to different branches are connected to each other, hence creating structures, called *formations*, for which we can ensure certain properties regarding the intersection between tree and path edges. Further, different formations are connected to each other by path edges in such a way to create bigger structures, called *extended formations*, which are, in their turn, connected to create a *sequence of extended formations*.

All of these structures are constructed in such a way that there exists a set of cells such that any four of its cells, connected to the same joint and being part of the same formation or extended formation, contain a region-level

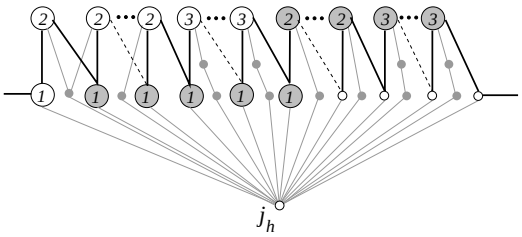


Figure 8.3: A cell. B-vertices of the head are depicted by large white circles, B-vertices of the tail are large grey circles, B-vertices not part of the cell (showing the tree structure) are small grey circles and stabilizers are small white cirlces. Tree edges are grey and path edges are black.

nonplanar tree for any possible leveling, where the levels correspond to cells. Hence, proving that four of such cells lie in different regions satisfying the properties of separation described above is equivalent to proving the existence of a crossing in the tree. This allows us to consider only the bigger structures instead of dealing with single copies of the region-level nonplanar tree.

In the following we define such structures more formally and state their properties.

Cell: The most basic structure defined by $\mathcal{P}$ is defined by looking at how it connects vertices of a set of branches connected to the same joint of $\mathcal{T}$. Assume the vertices of a level inside each branch to be arbitrarily ordered.

For each joint j_h, $h = 1, \ldots, q$, and for each disjoint subset of s branches B_i, $i = 1, \ldots, s$, connected to j_h, we construct a set of s cells as follows. For each $r = 1, \ldots, s$, a *cell* $c_r(h)$ is composed of its *head*, its *tail*, and a number t of stabilizers of j_h.

The *head* of $c_r(h)$ consists of the unique 1-vertex of B_r, the first three 2-vertices of each branch B_k, with $1 \leq k \leq s$ and $k \neq r$, that are not already used in a cell $c_a(h)$, with $1 \leq a < r$, and, for each 2-vertex not in $c_r(h)$ and not in B_r, the first 3-vertices not already used in a cell $c_a(h)$, with $1 \leq a < r$.

The *tail* of $c_r(h)$ consists of a set of $3 \cdot s \cdot (s-1)^2$ branches B_k adjacent to j_h. This set is partitioned into $3 \cdot (s-1)^2$ subsets of s subtrees each. The vertices of each of the subsets are distributed between the cells in the same way as for the vertices of the head.

This implies that each cell contains one 1-vertex, $3 \cdot (s-1)$ 2-vertices, and $3 \cdot (s-2) \cdot (s-1)$ 3-vertices of the head, an additional $3 \cdot (s-1)^2$ 1-vertices, $3^2 \cdot (s-1)^3$ 2-vertices, and $3^2 \cdot (s-2) \cdot (s-1)^3$ 3-vertices of the tail, plus

$3^2 \cdot (s-1)^4$ stabilizers.

Path $\mathcal{P}$ inside cell $c_r(h)$ visits the vertices in the following order: It starts at the unique 1-vertex of the head, then it reaches all the 2-vertices of the head, then all the 3-vertices of the head, then all the 2-vertices of the tail, and finally all the 3-vertices of the tail, visiting each set in arbitrary order. After each occurrence of a 2- or 3-vertex of the head, $\mathcal{P}$ visits a 1-vertex of the tail, and after each occurrence of a 2- or a 3-vertex of the tail, it visits a stabilizer of joint j_h (see Fig. 8.3).

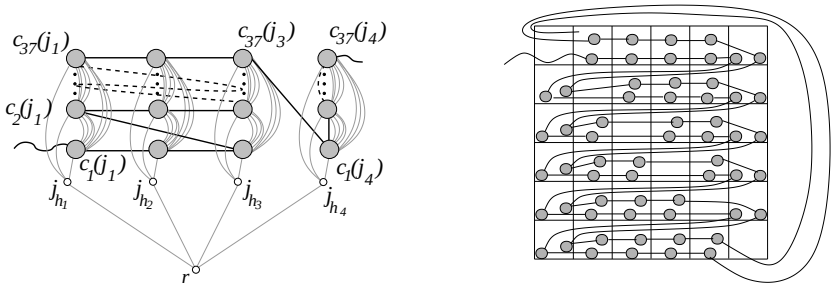
Note that, by this construction, each set of s cells connected to the same joint and constructed starting from the same set of s branches is such that each subset of size four contains region-level nonplanar trees with all possible levelings, where the levels correspond to the membership of the vertices to a cell. As an example, consider four arbitrary cells $c_1, \ldots, c_4$ belonging to the same set, leveled in this order. A region level nonplanar tree as in Fig. 8.1 consists of the 1-vertex v of the head of c_2, the three 2-vertices of c_3 connected to v and, for each of them, the 3-vertex of c_1 and the 3-vertex of c_4 connected to it. Other levelings are constructed analogously.

We now define two bigger structures describing how cells of this set are connected to cells of sets connected to other joints.

Formation: In the definition of a cell we described how the path traverses through one set of branches connected to the same joint. Now we describe how cells from four different sets are connected.

A *formation* $F(H)$, where $H = (h_1, h_2, h_3, h_4)$ is a 4-tuple of indices of joints, consists of 592 cells, namely of 148 cells $c_r(h_i)$ from one set of s cells as constructed above for each joint j_{h_i}, $1 \leq i \leq 4$. Path $\mathcal{P}$ connects these cells in the order $((h_1h_2h_3)^{37}h_4^{37})^4$, that is, $\mathcal{P}$ repeats four times the following sequence: It connects $c_1(h_1)$ to $c_1(h_2)$, then to $c_1(h_3)$, then to $c_2(h_1)$, and so on till $c_{37}(h_3)$, from which it then connects to $c_1(h_4)$, to $c_2(h_4)$, and so on till $c_{37}(h_4)$ (see Fig. 8.4(a)). A connection between two consecutive cells $c_r(a)$ and $c_r(b)$ is done with an edge connecting the end vertices of the parts $\mathcal{P}(c_r(a))$ and $\mathcal{P}(c_r(b))$ of $\mathcal{P}$ restricted to the vertices of $c_r(a)$ and $c_r(b)$, respectively. Namely, the unique vertex in $c_r(a)$ having degree 1 both in $\mathcal{P}(c_r(a))$ and in $\mathcal{T}$ is connected to the unique vertex in $c_r(b)$ having degree 1 in $\mathcal{P}(c_r(b))$ but not in $\mathcal{T}$. Since the cells in the formation that are connected to the same joint belong to the same set of s cells as constructed above, the following property holds:

Property 8.2 *For any formation $F(H)$ and any joint j_h, with $h \in H$, if four cells $c_r(h) \in F(H)$ are pairwise separated by straight lines, then there exists a*



(a) (b)

Figure 8.4: (a) A formation. Tree edges are depicted by grey and path edges by black lines. Please note in this figure also the bundle of tree edges connecting the different cells belonging to the same branch. (b) A subsequence $(H_1, \ldots, H_x)^2$ of an extended formation. Formations are inside a table to represent the 4-tuple they belong to and to emphasize that in each repetition (a row of the table) a formation at a certain 4-tuple is missing.

crossing in $\mathcal{T}$.

Extended Formation: Formations are connected by the path in a special sequence, defined as *extended formation* and denoted by $EF(H)$, where $H = (H_1 = (h_1, \ldots, h_4)$,
$H_2 = (h_5, \ldots, h_8), \ldots, H_x = (h_{4x-3}, \ldots h_{4x}))$ is a tuple of 4−tuples of disjoint indexes of joints (see Fig. 8.4(b)). Let $F_1(H_i), \ldots, F_{y-\frac{y}{x}}(H_i)$ be $y - \frac{y}{x}$ formations not belonging to any other extended formation and composed of cells of the same set S. These formations are connected in the order $(H_1, H_2, \ldots, H_x)^y$, but in each of these y repetitions one H_i is missing. Namely, in the k-th repetition the path does not reach any formation at H_m, with $m = k \mod x$. We say that the k-th repetition has a *defect* at m. We call a subsequence $(H_1, H_2, \ldots, H_x)^x$ a *full repetition* inside $EF(J)$. A full repetition has exactly one defect at each tuple.

Note that the size of s can now be fixed as the number of formations creating repetitions inside one extended formation times the number of cells inside each of these formations, that is $s := (y - \frac{y}{x}) \cdot 37 \cdot 4$. We claim that $x \leq 7 \cdot 3^2 \cdot 2^{23}$ and $y \leq 7^2 \cdot 3^3 \cdot 2^{26}$ is sufficient throughout the proofs. However, for readability reasons, we will keep on using variables x and y in the remainder of the chapter.

Sequence of Extended Formation: Extended formations are connected

by the path in a special sequence, called *sequence of extended formations* and denoted by $SEF(H)$, where $H = (H_1^*, \ldots, H_{12}^*)$ is a 12−tuple of tuples of 4−tuples. For each tuple H_i^*, where $i = 1, \ldots, 12$, consider 110 extended formations $(EF_i(H_1^*), \ldots, EF_i(H_{12}^*))$, with $i = 1, \ldots, 110$, not already belonging to any other sequence of extended formations. These extended formations are connected inside $SEF(H)$ in the order $(H_1^*, \ldots,$
$H_{12}^*)^{(120)}$. There exist two types of sequences of extended formations. Namely, in the first type there is one extended formation missing in each subsequence $(H_1^*, \ldots, H_{12}^*)$, that we call *defect*, as for the extended formations. In the second type, two consecutive extended formations are missing. Namely, in the k-th repetition the path skips the extended formations connecting at H_m^* and at H_{m+1}^*, with $m = k \bmod 12$. In this case, we say that the repetition has a *double defect*.

Since, for each set of $48x$ joints, $(48x)!$ different disjoint sequences of extended formations exist, we just consider the sequences where the order defined by the tuple is the order of the joints around the root.

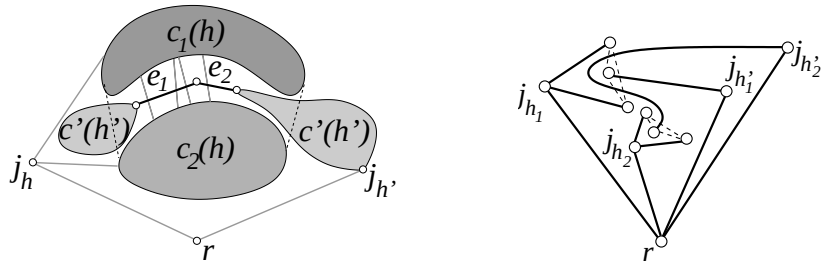
8.4 Overview

In this section we present the main arguments leading to the final conclusion that the tree $\mathcal{T}$ and the path $\mathcal{P}$ described in Sect. 8.3 do not admit any geometric simultaneous embedding. The main idea in this proof scheme is to use the structures given by the path to fix a part of the tree in a specific shape creating specific restrictions for the placement of the further substructures of $\mathcal{T}$ and of $\mathcal{P}$ attached to it.

We first give some further definitions and basic topological properties on the interaction among cells that are enforced by the preliminary arguments about region-level planar drawings and by the order in which the subtrees are connected inside one formation.

Passage: Consider two cells $c_1(h), c_2(h)$ that can not be separated by a straight line and a cell $c'(h')$, with $h' \neq h$. We say that there exists a *passage* P between c_1, c_2, and c' if the polyline given by the path of c' separates vertices of c_1 from vertices of c_2 (see Fig. 8.5(a)). Since the polyline can not be straight, there is a vertex of c' lying inside the convex hull of the vertices of $c_1 \cup c_2$, which implies the following.

Property 8.3 *In a passage between cells c_1, c_2, and c' there exist at least two path-edges e_1, e_2 of c' such that both e_1 and e_2 are intersected by tree-edges connecting vertices of c_1 to vertices of c_2.*

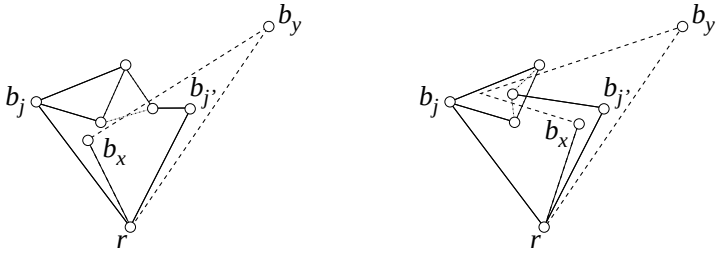


(a) (b)

Figure 8.5: (a) A passage between cells $c_1(h)$, $c_2(h)$, and $c'(h')$. (b) Two interconnected passages.

For two passages P_1 between $c_1(h_1)$, $c_2(h_1)$, and $c'(h'_1)$, and P_2 between $c_3(h_2)$, $c_4(h_2)$, and $c'(h'_2)$ (w.l.o.g., we assume $h_1 < h'_1$, $h_2 < h'_2$, and $h_1 < h_2$), we distinguish three different configurations: (i) If $h'_1 < h_2$, P_1 and P_2 are *independent*; (ii) if $h'_2 < h'_1$, P_2 is *nested* into P_1; and (iii) if $h_2 < h'_1 < h'_2$, P_1 and P_2 are *interconnected* (see Fig. 8.5(b)).

Doors: Let $c_1(h), c_2(h)$, and $c'(h')$ be three cells creating a passage. Consider any triangle given by a vertex v' of c' inside the convex hull of $c_1 \cup c_2$ and by any two vertices of $c_1 \cup c_2$. This triangle is a *door* if it encloses neither any other vertex of c_1, c_2 nor any vertex of c' that is closer than v' to $j_{h'}$ in $\mathcal{T}$. A door is *open* if no tree edge incident to v' crosses the opposite side of the triangle, that is, the side between the vertices of c_1 and c_2 (see Fig. 8.6(a)), otherwise it is *closed* (see Fig. 8.6(b)).



(a) (b)

Figure 8.6: (a) An open door. (B) A closed door.

Consider two joints j_a and j_b, with h, a, h', b appearing in this circular order around the root. Any polyline connecting the root to j_a, then to j_b, and again to the root, without crossing tree edges, must traverse each door by crossing both the sides adjacent to v'. If a door is closed, such a polyline has to bend after crossing one side adjacent to v' and before crossing the other one. Also, if two passages P_1 and P_2 are interconnected, either all the closed doors of P_1 are traversed by a path of tree-edges belonging to P_2 or all the closed doors of P_2 are traversed by a path of tree-edges belonging to P_1 (see Fig. 8.5(b)).

In the rest of the argument we will exploit the fact that the closed door of a passage requires a bend in the tree to obtain the claimed property that a large part of $\mathcal{T}$ has to follow the same shape. In view of this, we state the following lemmata relating the concepts of doors, passages, and formations.

Lemma 8.2 *For each formation $F(H)$, with $H = (h_1, \ldots, h_4)$, there exists a passage between some cells $c_1(h_a), c_2(h_a), c'(h_b) \in F(H)$, with $1 \leq a, b \leq 4$.*

Lemma 8.3 *Each passage contains at least one closed door.*

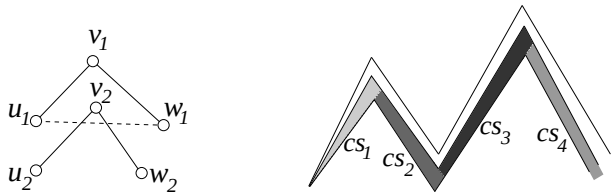
From the previous lemmata we conclude that each formation contains at least one closed door. To prove that the effects of closed doors belonging to different formations can be combined to obtain more restrictions on the way in which the tree has to bend, we exploit a combinatorial argument based on the Ramsey Theorem [GRS90] and state that there exists a set of joints pairwise creating passages.

Lemma 8.4 *Given a set of joints $J = \{j_1, \ldots, j_y\}$, with $|J| = y := \binom{2^7 \cdot 3 \cdot x + 2}{3}$, there exists a subset $J' = \{j'_1, \ldots, j'_r\}$, with $|J'| = r \geq 2^7 \cdot 3 \cdot x$, such that for each pair of joints $j'_i, j'_h \in J'$ there exist two cells $c_1(i), c_2(i)$ creating a passage with a cell $c'(h)$.*

Now we formally define the claimed property that part of the tree has to follow a fixed shape by considering how the drawing of the subtrees attached to two different joints force the drawing of the subtrees attached to the joints between them in the order around the root.

Enclosing bendpoints: Consider two paths $p_1 = \{u_1, v_1, w_1\}$ and $p_2 = \{u_2, v_2, w_2\}$. The bendpoint v_1 of p_1 *encloses* the bendpoint v_2 of p_2 if v_2 is internal to triangle $\triangle(u_1, v_1, w_1)$. See Fig. 8.7(a).

Channels: Consider a set of joints $J = \{j_1, \ldots, j_k\}$ in clockwise order around the root. The *channel* c_i of a joint j_i, with $i = 2, \ldots, k-1$, is the



(a) (b)

Figure 8.7: (a) An enclosing bendpoint. (b) A 3-channel and its channel segments.

region given by the pair of paths, one path of j_{i-1} and one path of j_{i+1}, with the maximum number of enclosing bendpoints with each other. We say that c_i is an *x-channel* if the number of enclosing bendpoints is x. Observe that, by Property 8.1, $x \leq 3$. A 3-channel is depicted in Fig. 8.7(b). Note that, given an x-channel c_i of j_i, all the vertices of the subtree rooted at j_i that are at distance at most x from the root lie inside c_i.

Channel segments: An x-channel c_i is composed of $x+1$ parts called *channel segments* (see Fig. 8.7(b)). The first channel segment cs_1 is the part of c_i that is visible from the root. The h-th channel segment cs_h is the region of c_i disjoint from cs_{h-1} that is bounded by the elongations of the paths of j_{i-1} and j_{i+1} after the h-th bend.

Observe that, as the channels are created by tree-edges, any tree-edge connecting vertices in the channel has to be drawn inside the channel, while path-edges can cross other channels. In the following we study the relationships between path-edges and channels.

The following property descends from the fact that every second vertex reached by $\mathcal{P}$ in a cell is either a 1-vertex or a stabilizer.

Property 8.4 *For any path edge $e = (a, b)$, at least one of a and b lie inside either cs_1 or cs_2.*

Blocking cuts: A *blocking cut* is a path edge connecting two consecutive channel segments by cutting some of the other channels twice. See Fig. 8.8.

Property 8.5 *Let c be a channel that is cut twice by a blocking cut. If c has vertices in both the channel segments cut by the path edge, then it has some vertices in a different channel segment.*

Proof: Consider the vertices lying in the two channel segments of c. In order to connect them in $\mathcal{T}$, a vertex v is needed in the bendpoint area of c. However, in order to have path connectivity between v and the vertices in the two channel segments, some vertices in a different channel segment are needed. □

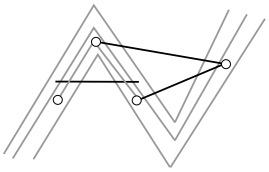


Figure 8.8: A blocking cut.

In the following lemma we show that in a set of joints as in Lemma 8.4 it is possible to find a suitable subset such that each pair of paths of tree-edges starting from the root and containing such joints has at least two common enclosing bendpoints, which implies that most of them create 2-channels.

Lemma 8.5 *Consider a set of joints $J = \{j_1, \ldots, j_k\}$ such that there exists a passage between each pair (j_i, j_h), with $1 \leq i, h \leq k$. Let $\mathcal{P}_1 = \{P \mid P$ connects c_i and $c_{\frac{3k}{4}+1-i}$, for $i = 1, \ldots, \frac{k}{4}\}$ and $\mathcal{P}_2 = \{P \mid P$ connects $c_{\frac{k}{4}+i}$ and c_{k+1-i}, for $i = 1, \ldots, \frac{k}{4}\}$ be two sets of passages between pairs of joints in J (see Fig. 8.17). Then, for at least $\frac{k}{4}$ of the joints of one set of passages, say $\mathcal{P}_1$, there exist paths in $\mathcal{T}$, starting at the root and containing these joints, which traverse all the doors of $\mathcal{P}_2$ with at least 2 and at most 3 bends. Also, at least half of these joints create an x-channel, with $2 \leq x \leq 3$.*

By Lemma 8.5, any formation attached to a certain subset of joints must use at least three different channel segments. In the remainder of the argument we focus on this subset of joints and give some properties holding for it, in terms of interaction between different formations with respect to channels. Since we need a full sequence of extended formations attached to these joints, k has to be at least eight times the number of channels inside a sequence of extended formations, that is, $k \geq 8 \cdot 48x = 2^7 \cdot 3x$.

First, we give some further definitions.

Nested formations A formation F is *nested* in a formation F' if there exist two edges $e_1, e_2 \in F$ and two edges $e'_1, e'_2 \in F'$ cutting a border cb of a channel c such that all the vertices of the path in F between e_1 and e_2 lie

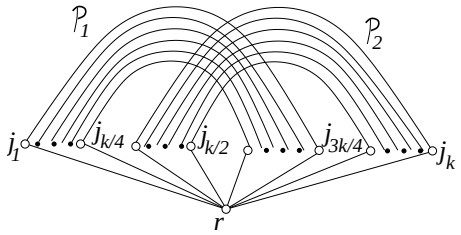
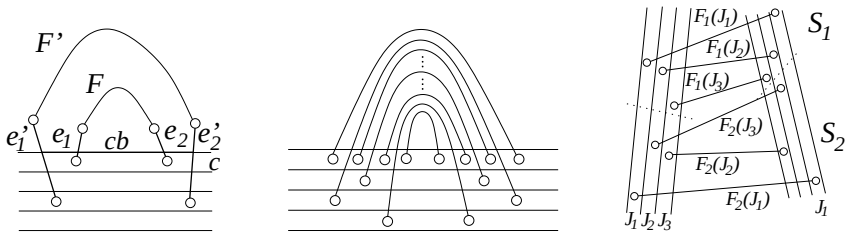


Figure 8.9: Two sets of passages $\mathcal{P}_1$ and $\mathcal{P}_2$ as described in Lemma 8.5.

inside the region delimited by cb and by the path in F' between e'_1 and e'_2 (see Fig. 8.10(a)).

A series of pairwise nested formations $F_1, \ldots, F_k$ is *r-nested* if there exist r formations $F_{q_1}, \ldots, F_{q_r}$, with $1 \leq q_1, \ldots, q_r \leq k$, belonging to the same channel and such that, for each pair $F_{q_p}, F_{q_{p+1}}$, there exists at least one formation F_z, $1 \leq z \leq k$, belonging to another channel and such that F_{q_p} is nested in F_z and F_z is nested in $F_{q_{p+1}}$ (see Fig. 8.10(b)).



(a) (b) (c)

Figure 8.10: (a) A formation F nested in a formation F'. (b) A series of r-nested formations. (c) Two independent sets S_1 and S_2.

Independent sets of formations Let $S_1, \ldots, S_k$ be sets of formations of one extended formation such that each set S_i contains formations $F_i(H_1), \ldots, F_i(H_r)$ on the set of 4-tuples $H = \{H_1, \ldots, H_r\}$, where the joints of H_i are between the joints of H_{i-1} and of H_{i+1} in the order around the root. Further, let $F_a(H_c)$ be not nested in $F_b(H_d)$, for each $1 \leq a, b \leq k$, $a \neq b$, and $1 \leq c, d \leq r$. If for each pair of sets S_a, S_b there exist two lines l_1, l_2 separating the vertices of S_a and S_b inside channel segment cs_1 and cs_2, respectively, the sets are *independent* (see Fig. 8.10(c)).

In the following lemmata we prove that in any extended formation there exists a nesting of a certain depth (Lemma 8.8). This important property will be the starting point for the final argument and will be deeply exploited in the rest of the chapter. We get to this conclusion by first proving that in an extended formation the number of independent sets of formations is limited (Lemma 8.6) and then by showing that, although there exist formations that are neither nested nor independent, in any extended formation there exists a certain number of pairs of formation that have to be either independent or nested (Lemma 8.7).

Lemma 8.6 *There exist no $n \geq 2^{22} \cdot 14$ independent sets of formations $S_1, \ldots, S_n$ inside any extended formation, where each S_i contains formations of a fixed set of channels of size $r \geq 22$.*

Lemma 8.7 *Consider four subsequences $Q_1, \ldots, Q_4$, where $Q_i = (H_1, H_2, \ldots, H_x)$, of an extended formation EF, each consisting of a whole repetition of EF. Then, there exists either a pair of nested subsequences or a pair of independent subsequences.*

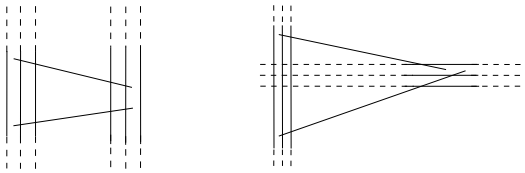
Lemma 8.8 *Consider an extended formation $EF(H_1, H_2, \ldots, H_x)$. Then, there exists a k-nesting, where $k \geq 6$, among the formations of EF.*

Once the existence of 2-channels and of a nesting of a certain depth in each extended formation has been shown, we turn our attention to study how such a deep nesting can be performed inside the channels.

Let cs_a and cs_b, with $1 \leq a, b \leq 4$, be two channel segments. If the elongation of cs_a intersects cs_b, then it is possible to connect from cs_b to cs_a by cutting both the sides of cs_a. In this case, cs_a and cs_b have a *2-side connection* (see Fig. 8.11(b)). On the contrary, if the elongation of cs_a does not intersect cs_b, only one side of cs_a can be used. In this case, cs_a and cs_b have a *1-side connection* (see Fig. 8.11(a)).

Based on these different ways of connecting distinct channel segments, we split our proof into three parts, the first one dealing with the setting in which only 1-side connections are allowed, the second one allowing one single 2-side connection, and the last one tackling the general case.

Proposition 8.1 *If there exist only 1-side connections, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*



(a) (b)

Figure 8.11: (a) A 1-side connection. (b) A 2-side connection.

We prove this proposition by showing that, in this configuration, the existence of a deep nesting in a single extended formation, proved in Lemma 8.8, results in a crossing in either $\mathcal{T}$ or $\mathcal{P}$.

Lemma 8.9 *If an extended formation lies in a part of the channel that contains only 1-side connections, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Next, we study the case in which there exist 2-side connections. We distinguish two types of 2-side connections, based on the fact that the elongation of channel segment cs_a intersecting channel segment cs_b starts at the bendpoint that is closer to the root, or not. In the first case we have a *low Intersection* (see Fig. 8.12(a)), denoted by $I^l_{(a,b)}$, and in the second case we have a *high Intersection* (see Fig. 8.12(b)), denoted by $I^h_{(a,b)}$, where $a, b \in \{1, \ldots, 4\}$. We use the notation $I_{(a,b)}$ to describe both $I^h_{(a,b)}$ and $I^l_{(a,b)}$. We say that two intersections $I_{(a,b)}$ and $I_{(c,d)}$ are *disjoint* if $a, d \in \{1, 2\}$ and $b, c \in \{3, 4\}$. For example, $I_{(1,3)}$ and $I_{(4,2)}$ are disjoint, while $I_{(1,3)}$ and $I_{(2,4)}$ are not.

(a) (b)

Figure 8.12: (a) A low Intersection. (b) A high Intersection.

Since consecutive channel segments can not create any 2-side connection, in order to explore all the possible shapes we consider all the combinations of low and high intersections created by channel segments cs_1 and cs_2 with channel segments cs_3 and cs_4. With the intent of proving that intersections of different channels have to maintain certain consistencies, we state the following lemma.

Lemma 8.10 *Consider two channels ch_p, ch_q with the same intersections. Then, none of channels ch_i, where $p < i < q$, have an intersection that is disjoint with the intersections of ch_p and of ch_q.*

As for Proposition 8.1, in order to prove that 2-side connections are not sufficient to obtain a simultaneous embedding of $\mathcal{T}$ and $\mathcal{P}$, we exploit the existence of the deep nesting shown in Lemma 8.8. First, we analyze some properties relating such nesting to channel segments and bending areas. A *bending area* $b(a, a+1)$ is the region between cs_a and cs_{a+1} where bendpoints can be placed. We first observe that all the extended formations have to place vertices inside the bending area of the channel segment where the nesting takes place, and then prove that not many of the formations involved in the nesting can use the part of the path that creates the nesting to place vertices in such a bending area, which implies that the extended formations have to reach the bending area in a different way.

Lemma 8.11 *Consider an x-nesting of a sequence of extended formations on an intersection $I_{(a,b)}$, with $a \leq 2$. Then, there exists a triangle t in the nesting that separates some of the triangles nesting with t from the bending area $b(a, a+1)$ (or $b(a-1, a)$).*

Then, we study some of the cases involving 2-side connections and we show that the connections between the bending area and the "endpoints" of the nesting create a further nesting of depth greater than 6. Hence, if no further 2-side connection is available, this second nesting is not drawable.

Proposition 8.2 *Let t be a triangle open on a side splitting a channel segment cs into two parts such that every extended formation EF has vertices in both parts. If the only possibility to connect vertices in different parts of cs is with a 1-side connection and if any such connection creates a triangle open on a side that is nested with t, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

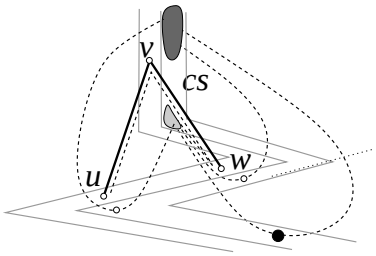


Figure 8.13: A situation as in Proposition 8.2. The chosen turning vertex is represented by a big black circle and is in configuration β. The inner and the outer areas are represented by a light grey and a dark grey region, respectively.

Refer to Fig. 8.13. Consider the two path-edges $e_1 = (u, v), e_2 = (v, w)$ creating t such that the common point v is in the channel segment cs that is split into two parts, that we call *inner area* and *outer area*, respectively. We assume that e_1, e_2 do not cut any channel segment cs' completely, since such a cut would create more restrictions than placing u or w inside cs'. Consider the path in an extended formation EF connecting the inner and the outer area through a 1-side connection at cs'. As a generalization, consider for such a path of EF only a vertex, called *turning vertex*, which is placed in cs' and for which no other path in EF exists that connects the inner and the outer area by using a channel segment cs'' such that the subpath to cs'' intersects either cs'' or its elongation. If there exist more than one of such vertices, then arbitrarily choose one of them. Observe that the path connecting from the inner area to the outer area through the turning vertex encloses exactly one of u and w. If it encloses u, it is in configuration α, otherwise it is in configuration β. If there exist both paths in α and paths in β configuration, then we arbitrarily consider one of them. Finally, consider the connections between different extended formations inside a sequence of extended formations. Consider a turning vertex v in a channel segment cs of a channel ch such that the edges incident to v cut a channel ch'. Then, any connection of an extended formation of ch' from the inner to the outer area in the same configuration as ch and with its turning vertex v' in cs is such that v' lies inside the convex hull of the two edges incident to v.

In the following two lemmata we show that in the setting described in Proposition 8.2 there exists a crossing either in $\mathcal{T}$ or in $\mathcal{P}$.

Lemma 8.12 *In a situation as described in Proposition 8.2, not all the extended formations in a sequence of extended formations can place turning vertices in the same channel segment.*

Lemma 8.13 *In a situation as described in Proposition 8.2, $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Based on the property given by Proposition 8.2, we present the second part of the proof, in which we show that having two intersections $I_{(a,b)}$ and $I_{(c,d)}$ does not help if $I_{(a,b)}$ and $I_{(c,d)}$ are not disjoint.

Proposition 8.3 *If there exists no pair of disjoint* 2*-side connections, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Observe that, in this setting, it is sufficient to restrict the analysis to cases $I_{(1,3)}$ and $I_{(3,1)}$, since the cases involving 2 and 4 can be reduced to them.

Lemma 8.14 *If a shape contains an intersection $I_{(1,3)}$ and does not contain any other intersection that is disjoint with $I_{(1,3)}$, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Lemma 8.15 *If there exists a sequence of extended formation in any shape containing an intersection $I_{(3,1)}$, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Observe that, in the latter lemma, we proved a property that is stronger than the one stated in Proposition 8.3. In fact, we proved that a simultaneous embedding cannot be obtained in any shape containing an intersection $I_{(3,1)}$, even if a second intersection that is disjoint with $I_{(3,1)}$ is present.

Finally, in the third part of the proof, we tackle the general case where two disjoint intersections exist.

Proposition 8.4 *If there exists two disjoint* 2*-side connections, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Since the cases involving intersection $I_{3,1}$ were considered in Lemma 8.15, we only have to consider the eight different configurations where one intersection is $I_{(1,3)}$ and the other is one of $I_{(4,\{1,2\})}$. In the next three lemmata we cover the cases involving $I^h_{(1,3)}$ and in Lemma 8.19 the ones involving $I^l_{(1,3)}$.

Consider two consecutive channel segments cs_i and cs_{i+1} of a channel c and let e be a path-edge crossing the border of one of cs_i and cs_{i+1}, say cs_i. We say that e creates a *double cut* at c if the elongation of e cuts c in cs_{i+1}. A double cut is *simple* if e does not cross cs_{i+1} (see Fig. 8.14(a)) and *non-simple* otherwise (see Fig. 8.14(b)). Also, a double cut of an extended formation EF is *extremal* with respect to a bending area $b(x, x+1)$ if there exists no double cut of EF that is closer than it to $b(x, x+1)$.

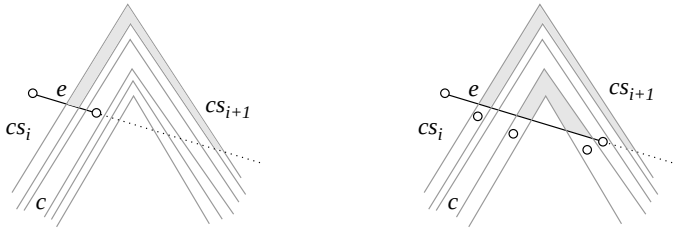


Figure 8.14: (a) A simple double cut. (b) A non-simple double cut.

Property 8.6 *Any edge e_k creating a double cut at a channel k in channel segment cs_i blocks visibility to the bending area $b(i, i+1)$ for a part of cs_i in each channel ch_h with $h > k$ (with $h < k$).*

In the following lemma we show that a particular ordering of extremal double cuts in two consecutive channel segments leads to a non-planarity in $\mathcal{T}$ or $\mathcal{P}$. Note that, any order of extremal double cuts corresponds to an order of the connections of a subset of extended formations to the bending area.

Lemma 8.16 *Let cs_i and cs_{i+1} be two consecutive channel segments. If there exists an ordered set $S := (1, 2, \ldots, 5)^3$ of extremal double cuts cutting cs_i and cs_{i+1} such that the order of the intersections of the double cuts with cs_i (with cs_{i+1}) is coherent with the order of S, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Then, we show that shape $I^h_{(1,3)}\ I_{(4,2)}$ induces this order. To prove this, we first state the existence of double cuts in shape $I^h_{(1,3)}\ I^h_{(4,2)}$. The existence of double cuts in shape $I^h_{(1,3)}\ I^l_{(4,2)}$ can be easily seen.

Lemma 8.17 *Each extended formation in shape $I^h_{(1,3)}$ $I^h_{(4,2)}$ creates double cuts in at least one bending area.*

Lemma 8.18 *Every sequence of extending formations in shape $I^h_{(1,3)}$ $I^{h,l}_{(4,2)}$ contains an ordered set $(1, 2, \ldots, 5)^3$ of extremal double cuts with respect to bending area either $b(2, 3)$ or $b(3, 4)$.*

Finally, we consider the configurations where one intersection is $I^l_{(1,3)}$ and the other one is one of $I^{h,l}_{(4,2)}$. Observe that, in both cases, channel segment cs_2 is on the convex hull.

Lemma 8.19 *If channel segment cs_2 is part of the convex hull, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Based on the above discussion, we state the following theorem.

Theorem 8.1 *There exist a tree and a path that do not admit any geometric simultaneous embedding.*

Proof: Let $\mathcal{T}$ and $\mathcal{P}$ be the tree and the path described in Sect. 8.3. Then, by Lemma 8.5, Lemma 8.10, and Property 8.1, a part of $\mathcal{T}$ has to be drawn inside channels having at most four channel segments. Also, by Lemma 8.8, there exists a nesting of depth at least 6 inside each extended formation.

By Proposition 8.1, if there exist only 1-side connections, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any simultaneous embedding. By Proposition 8.3, if there exists either one 2-side connections or a pair of non-disjoint intersections, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any simultaneous embedding. By Proposition 8.4, even if there exist two disjoint 2-side intersections, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any simultaneous embedding. Since it is not possible to have more than two disjoint 2-side intersections, the statement follows. □

8.5 Detailed Proofs

Lemma 8.2. *For each formation $F(H)$, with $H = (h_1, \ldots, h_4)$, there exists a passage between some cells $c_1(h_a), c_2(h_a), c'(h_b) \in F(H)$, with $1 \leq a, b \leq 4$.*

Proof: Suppose, for a contradiction, that there exists no passage inside $F(H)$. First observe that, if two cells $c_1(h_a), c_2(h_a) \in F(H)$ are separated by a polyline given by the path passing through $F(H)$, then either they are separable by a straight line or such a polyline is composed of edges belonging

to a cell $c_3(h_a)$ of the same joint j_{h_a}. Since, by Property 8.2, there exists no set of four cells of a given joint inside $F(H)$ that are separable by a straight line, all the cells of $F(H)$ of a given joint can be grouped into at most 3 different sets S^1, S^2, and S^3 such that cells from different sets can be separated by straight lines, but cells from the same set can not. Therefore, the cells inside one of these sets can only be separated by other cells of the same set.

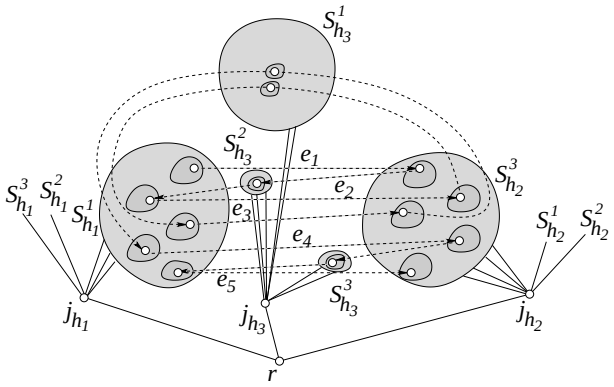


Figure 8.15: The five path edges $e_1, \ldots, e_5$ connecting five cells of set $S^a_{h_1}$ with five cells of set $S^b_{h_2}$.

Consider the connections of the path through $F(H)$ with regard to this notion of sets of cells. Let $S^y_{h_x}$, with $x = 1, \ldots, 4$ and $y = 1, \ldots, 3$, be the set of cells belonging to set S^y and attached to joint j_{h_x}. Hence, for any two cells $c_1(h_x), c_2(h_{x+1})$ there are nine possible ways to connect between some $S^y_{h_x}$ and $S^{y'}_{h_{x+1}}$. Since the part of $\mathcal{P}$ through $F(H)$ visits 37 times cells from $j_{h_1}, j_{h_2}, j_{h_3}$, in this order, there exist five path edges $e_1, \ldots, e_5$ connecting five cells of set $S^a_{h_1}$ with five cells of set $S^b_{h_2}$, where $1 \leq a, b \leq 3$ (see Fig.8.15). Without loss of generality, we assume that edges $e_1, \ldots, e_5$ appear in this order in the part of $\mathcal{P}$ through $F(H)$. Observe that $e_1, \ldots, e_5$, together with the five cells of $S^a_{h_1}$ and the five cells of $S^b_{h_2}$ they connect, subdivide the plane into five regions. Since the path is continuous in $F(H)$, it connects from the end of e_1 (a cell of joint j_{h_2}) to the beginning of e_2 (a cell of joint j_{h_1}), from the end of e_2 to the beginning of e_3, and so on. If in the region between edges e_s and e_{s+1}, with $1 \leq s \leq 4$, there exists no cell of joint j_{h_3}, then the path through $F(H)$

will not traverse the region between these edges in the opposite direction, since the path contains no edges going from a cell of j_{h_2} to a cell of j_{h_1} and since the start- (and end-) cells of these edges cannot be separated by straight lines. Furthermore, note that, in this case, the path-connection from e_s to e_{s+1} does not traverse the region between the edges, therefore forming a spiral shape, in the sense that the part of the path following e_{s+1} is separated from the part of the path prior to e_s. Since we have five edges between $S^a_{h_1}$ and $S^b_{h_2}$ but only 3 possible sets of cells on joint j_{h_3}, at least one pair of edges exists creating an empty region and therefore a spiral separating the path.

By this argument, it follows that cells attached to joint j_{h_4} in different repetitions of the subsequence $((h_1h_2h_3)^{37}h_4^{37})$ in $F(H)$ are separated by path edges of the spirals formed by the repeated subsequence of visited cells of the joints $j_{h_1}, j_{h_2}, j_{h_3}$. Since four repetitions create four of such separated cells on j_{h_4}, by Property 8.2 there exists a pair of cells that are not separable by a straight line but are separated by the path. Since the path of the spiral separating them consists only of cells on different joints, any possible separating polyline leads to a contradiction to the non-existence of a passage inside $F(H)$. □

Lemma 8.3. *Each passage contains at least one closed door.*

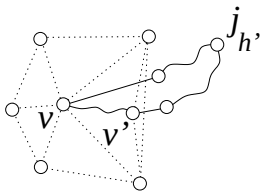


Figure 8.16: There exists a closed door in each passage.

Proof: Refer to Fig. 8.16. Let P_1 be a passage between $c_1(h)$, $c_2(h)$, and $c'(h')$. Consider any vertex v of c' inside the convex hull of $C := c_1 \cup c_2$. Further, consider all the triangles $\triangle(v, v_1, v_2)$ created by v with any two vertices $v_1, v_2 \in C$ such that $\triangle(v, v_1, v_2)$ does not enclose any other vertex of C. The path of tree edges connecting v to $j_{h'}$ enters one of the triangles. Then, either it leaves the triangle on the opposite side, thereby creating a closed door, or it encounters a vertex v' of c'. Since at least one vertex of c' lies outside the convex hull of C, otherwise they would not be separated by c', it is possible to repeat the argument on triangle $\triangle(v', v_1, v_2)$ until a closed door is found. □

Lemma 8.4. *Given a set of joints $J = \{j_1, \ldots, j_y\}$, with $|J| = y := \binom{2^7 \cdot 3 \cdot x + 2}{3}$, there exists a subset $J' = \{j'_1, \ldots, j'_r\}$, with $|J'| = r \geq 2^7 \cdot 3 \cdot x$, such that for each pair of joints $j'_i, j'_h \in J'$ there exist two cells $c_1(i), c_2(i)$ creating a passage with a cell $c'(h)$.*

Proof: By construction of the tree, for each set of four joints, there are formations that visit only cells of these joints. By Lemma 8.2, there exists a passage inside each of these formations, which implies that for each set of four joints there exists a subset of two joints creating a passage. The actual number of joints needed to ensure the existence of a subset of joints of size r such that passages exist between each pair of joints is given by the Ramsey Number $R(r, 4)$. This number is defined as the minimal number of vertices of a graph G such that G either has a complete subgraph of size r or an independent set of size 4. Since in our case we can never have an independent set of size 4, we conclude that a subset of size r exists with the claimed property. The Ramsey number $R(r, 4)$ is not exactly known, but we can use the upper bound directly extracted from the proof of the Ramsey theorem to arrive at the bound stated above. [GRS90] □

Lemma 8.5. *Consider a set of joints $J = \{j_1, \ldots, j_k\}$ such that there exists a passage between each pair (j_i, j_h), with $1 \leq i, h \leq k$. Let $\mathcal{P}_1 = \{P \mid P$ connects c_i and $c_{\frac{3k}{4}+1-i}$, for $i = 1, \ldots, \frac{k}{4}\}$ and $\mathcal{P}_2 = \{P \mid P$ connects $c_{\frac{k}{4}+i}$ and c_{k+1-i}, for $i = 1, \ldots, \frac{k}{4}\}$ be two sets of passages between pairs of joints in J (see Fig. 8.17). Then, for at least $\frac{k}{4}$ of the joints of one set of passages, say $\mathcal{P}_1$, there exist paths in $\mathcal{T}$, starting at the root and containing these joints, which traverse all the doors of $\mathcal{P}_2$ with at least 2 and at most 3 bends. Also, at least half of these joints create an x-channel, with $2 \leq x \leq 3$.*

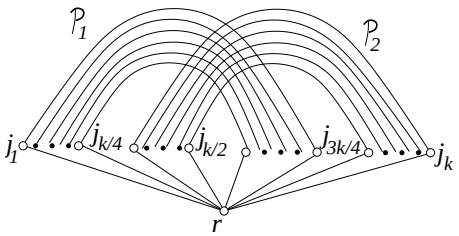
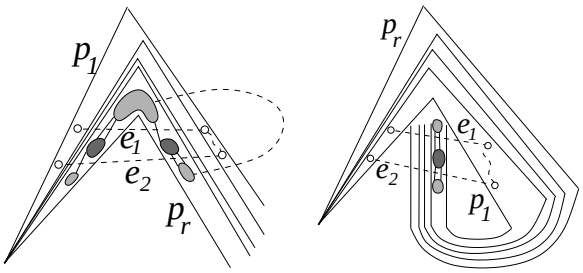


Figure 8.17: The two sets of passages $\mathcal{P}_1$ and $\mathcal{P}_2$ described in Lemma 8.5.

Proof: Observe first that each passage of $\mathcal{P}_1$ is interconnected with each

passage of $\mathcal{P}_2$ and that all the passages of $\mathcal{P}_1$ and all the passages of $\mathcal{P}_2$ are nested.

By Lemma 8.3 and Property 8.1, for one of $\mathcal{P}_1$ and $\mathcal{P}_2$, say $\mathcal{P}_1$, either for every joint of $\mathcal{P}_1$ between the joints of $\mathcal{P}_2$ in the order around the root or for every joint of $\mathcal{P}_1$ not between the joints of $\mathcal{P}_2$, there exists a path p_i in $\mathcal{T}$, starting at the root and containing these joints, which has to traverse all the doors of $\mathcal{P}_2$ by making at least 1 and at most 3 bends. Also, paths $p_1, \ldots, p_{\frac{k}{4}}$ can be ordered in such a way that a bendpoint of p_i encloses a bendpoint of p_h for each $h > i$. It follows that there exist x-channels with $1 \leq x \leq 3$ for each joint. Consider now the set of joints $J' \subset J$ visited by these paths. We assume the joints of $J' = \{j'_1, \ldots j'_r\}$ to be in this order around the root.



(a) (b)

Figure 8.18: (a) The separating cell c' is in the outermost channel. (b) The separating cell c' is in the innermost channel.

Consider the path p_1 whose bendpoint encloses the bendpoint of each of all the other paths and the path p_r whose bendpoint encloses the bendpoint of none of the other paths (see Figs. 8.18(a) and 8.18(b)). Please note that either p_1 visits j'_1 and p_r visits j'_r or vice versa, say p_1 visits j'_1. By construction, there exists a passage between cells from j'_1 and cells from j'_r. In this passage there exist either two path-edges e_1, e_2 of a cell $c'(1)$ separating two cells $c_1(r), c_2(r)$, thereby crossing the channel of j'_r, or two edges of a cell $c'(r)$ separating two cells $c_1(1), c_2(1)$, thereby crossing the channel of j'_1. We show that 1-channels are not sufficient to draw these passages.

In the first case (see Fig. 8.18(a)), both separating edges e_1, e_2 cross the path p_r before and after the bend, thereby creating blocking cuts separating vertices of the same cell, say c_1. Since they are connected by the path, by Property 8.5, an additional bend is needed. In the other case (see Fig. 8.18(b)),

any edge connecting vertices of $c'(j'_r)$ is not even crossing any edge of p_1 and therefore at least another bend is needed in the channel. So at least one of the joints needs an additional bend. Since there are passages between each pair of joints in J', all but one joint j_q have a path that has to bend an additional time. We note that the additional bendpoint of each path p_k aside from p_1, p_r, and p_q has to enclose all the additional bendpoints either of $p_1, \ldots, p_{k-1}$ or of $p_{k+1}, \ldots, p_r$. It follows that, for at least half of the joints, there exist x-channels where $2 \leq x \leq 3$. □

Lemma 8.6. *There exist no $n \geq 2^{22} \cdot 14$ independent sets of formations $S_1, \ldots, S_n$ inside any extended formation, where each S_i contains formations of a fixed set of channels of size $r \geq 22$.*

Proof: Assume for a contradiction, that such independent sets $S_1, \ldots, S_n$ exist. By Lemma 8.2, each formation in each set will contain a passage and thereby an edge cutting the channel border. By Property 8.4 each formation in each set will place an edge to either channel segment cs_1 or cs_2. As can be easily seen, there exists a set S^1 of size $\frac{n}{2}$ of sets of formations that will have at least one common connection for a fixed formation F_i in each set $S_a \subset S^1$, where $1 \leq n$. By repeating the argument we can find a subset $S^2 \subset S^1$ of size $\frac{n}{4}$ such that these sets will have at least two common connections for formations F_i, F_h in each set $S_a \subset S^2$. By continuing this procedure we arrive at a subset S^r of size $\frac{n}{2^r}$ that will have at least r common connections. Since all these common connections have to connect to either cs_1 or cs_2, we have identified a set $S = \{S'_1, \ldots, S'_{\frac{n}{2^r}}\}$ of size $\frac{n}{2^r}$ of sets of formations of size at least $\frac{r}{2}$ that has all the connections to one specific channel segment CS.

We now consider the cutting edges for each of the formations of S in CS. Since any of those can intersect the channel border on two different sides, at least half of the connections for a fixed formation $F_{\frac{r}{4}}$ in all the sets will intersect with one side of the channel border, thereby crossing either all the channels $1, \ldots, \frac{r}{4} - 1$ or all the channels $\frac{r}{4} + 1, \ldots, \frac{r}{2}$, assume the first. Consider now the formations $F_{\frac{r}{8}}$ in each of the sets. These formations of the sets $S'_2, S'_4, \ldots, S'_{\frac{n}{2^{r+1}}}$ will be separated on CS by the edges of the formations $F_{\frac{r}{4}}$ of the sets $S'_3, S'_5, \ldots, S'_{\frac{n}{2^r}-1}$. To avoid a monotonic ordering of the separated formations and thereby the existence of an region-level nonplanar tree these formations $F_{\frac{r}{8}}$ have to place vertices in an adjacent channel segment CS'. This will create blocking cuts for either all the channels $1, \ldots, \frac{r}{8} - 1$ or all the channels $\frac{r}{8} + 1, \ldots, \frac{r}{4}$, assume the first. Consider now the formations F_1 in each of the sets. These formations of the sets $S'_3, S'_5, \ldots, S'_{\frac{n}{2^r}-2}$ will be separated on CS by the edges of the formations $F_{\frac{r}{8}}$ of the sets $S'_4, S'_6, \ldots, S'_{\frac{n}{2^r}-3}$. By the

same argument as above also these formations have to place vertices in an adjacent channel segment that are visible from some of the separated areas of CS. Since the connection from the formations $F_{\frac{r}{8}}$ are blocking for the connection to CS', the formations F_1 have to use the remaining adjacent channel segment CS'', thereby blocking all the channels $1, \ldots r_2$. We finally consider the formations F_2 of the sets $S'_4, S'_6, \ldots, S'_{10}$. These formations are now separated in CS by the blocking edges to CS' of the formations $F_{\frac{r}{8}}$ and by the blocking edges to CS'' of the formations F_1. Therefore, these formations cannot use part of any channel segment (tree-)visible to the separated areas in CS. So, by Property 8.2, we identified a region-level nonplanar tree, in contradiction to the assumption. □

Lemma 8.7. *Consider four subsequences $Q_1, \ldots, Q_4$, where $Q_i = (H_1, H_2, \ldots, H_x)$, of an extended formation EF, each consisting of a whole repetition of EF. Then, there exists either a pair of nested subsequences or a pair of independent subsequences.*

Proof: Assume that no pair of nested subsequences exists. We show that a pair of independent subsequences exists.

First, we consider how $Q_1, \ldots, Q_4$ use the first two channel segments cs_1 and cs_2. Each of these subsequences uses either only cs_1, only cs_1, or both to place its formations. Observe that, if a subsequence uses only cs_1 and another one uses only cs_2, then such subsequences are clearly independent. So we can assume that all of $Q_1, \ldots, Q_4$ use a common channel segment, say cs_2.

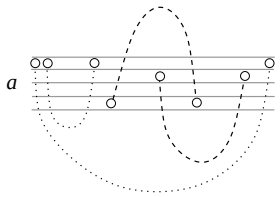


Figure 8.19: If three subsequences use the same channel segment cs, then at least two of them are either nesting or separated in cs.

Then we show that, if three subsequences use the same channel segment cs, then at least two of them are separated in cs. In fact, if two subsequences using cs are not independent, then they contain formations on the same channel a that intersect with different channel borders of a. However, a third subsequence containing a formation that intersects a channel border of a is such that there

exists either a nesting or a clear separation between this subsequence and the other subsequence intersecting the same channel border of a (see Fig. 8.19). This fact implies that if three subsequences use only cs_2, then at least two of them are independent. From this and from the fact that there are four subsequences using cs_2, we derive that two subsequences, say Q_1, Q_2, are separated in cs_2 and are not separated in cs_1. Then, the third subsequence Q_3 can be placed in such a way that it is not separated from Q_1 and Q_2 in cs_2. However, this implies that Q_4 is separated in cs_1 from two of Q_1, Q_2, Q_3 and in cs_2 from two of Q_1, Q_2, Q_3, which implies that Q_4 is separated in both channel segments from one of Q_1, Q_2, Q_3. □

Lemma 8.8. *Consider an extended formation $EF(H_1, H_2, \ldots, H_x)$. Then, there exists a k-nesting, where $k \geq 6$, among the formations of EF.*

Proof: Assume, for a contradiction, that there is no k-nesting among the sequence of formations in EF. We claim that, under this assumption, there exist more than n sequences of independent formations in EF from the same set of channels C, where $n \geq 2^{22} \cdot 14$ and $|C| \geq 22$. By Lemma 8.6, such a claim clearly implies the statement.

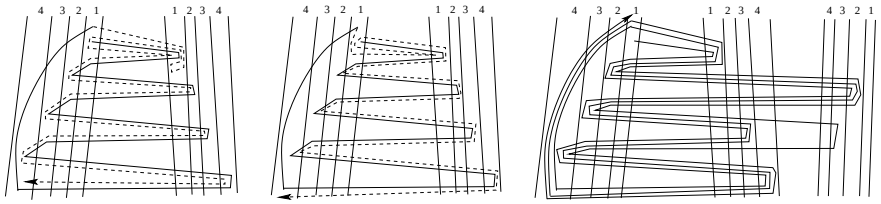
Consider sequences that use some common channels in channel segments cs_1 and cs_2. Then, their separation in cs_1 has the opposite ordering with respect to their separation in cs_2.

Observe that, by Lemma 8.7, there exist at most $(n-1) \cdot 3$ different nestings of subsequences such that there are less than n independent sets of subsequences. Also note that, if some formations belonging to two different subsequences are nesting, then all the formations of these subsequences have to be part of some nesting. However, this does not necessarily mean for all the formations to nest with each other and to build a single nesting.

Since the number of channels used inside EF is greater than $(n-1) \cdot 3 \cdot 3$, where $n \geq 2^{22} \cdot 14$, we have a nesting consisting of subsequences with at least 3 different defects.

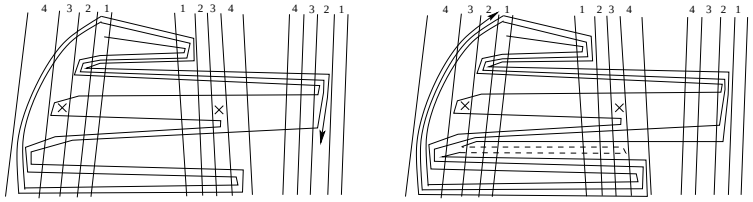
Let the nesting consist of subsequences $Q_1^1, \ldots, Q_1^r, Q_2^1, \ldots, Q_2^r, \ldots, Q_k^1 \ldots, Q_k^r$, where Q_i^h denotes the h-th occurrence of a subsequence of EF with a defect at channel i. Further, let the path connect them in the order $Q_1^1, Q_2^1, \ldots, Q_k^1, Q_1^2, \ldots, Q_k^2, \ldots, Q_k^r$. We show that there exists a pair of independent subsequences within this nesting.

Consider now the first two nesting repetitions of sequence $(H_1, H_2, \ldots, H_x)$, that is, Q_1^1 and Q_2^1. Let the nesting consist of a formation $F(k)$ from Q_1^1 nesting in a formation $F'(s)$ from Q_2^1. Consider the edges $e_1, e_2 \in F(k)$ and $e_1', e_2' \in F'(s)$ that are responsible for the nesting. Without loss of generality



(a) (b) (c)

Figure 8.20: (a) and (b) Possible configurations for Q_1^1, Q_2^1, and Q_3^1. (c) The repetitions follow the outward orientation.



(a) (b)

Figure 8.21: The connection between channels 2 and 4 blocks visibility for the following repetitions to the part of the channel segment where vertices of channel 3 were placed till that repetition.

we assume the path p that connects e_2' and e_1 not to contain edges e_1', e_2. Consider the two parts a, b of the channel border of s, where a is between e_1 and e_1' and b is between e_2 and e_2'. Consider now the closed region delimited by the path through $F'(s)$, the path p, the path through $F(k)$, and b. Such a region is split into two closed regions R_{in} and R_{nest} by a (see Fig. 8.23).

Observe that, in order to reach from R_{in} to the outer region, any path has to cross both a and b. We note that the part of $\mathcal{P}$ starting at e_2' and not containing $F(k)$ is either completely contained in the outer region or has to cross over between R_{in} and the outer region by traversing R_{nest}. Similarly, the the part of $\mathcal{P}$ starting at e_1 and not containing $F'(s)$ either does not reach the outer region or has to cross over between R_{in} and the outer region by traversing R_{nest}. Furthermore, any formation F'' on such a path is also either crossing

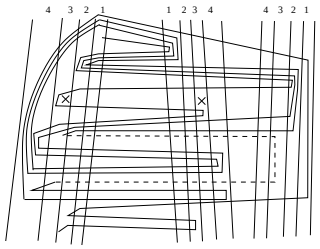


Figure 8.22: All the channels $c, \ldots, x$ are shifted and the next repetition starts in a completely different region.

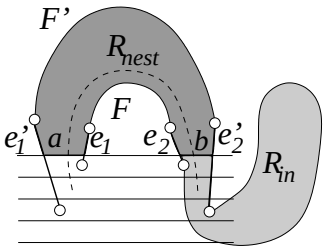


Figure 8.23: Regions R_{in} and R_{nest}.

over and thereby cutting both a and b, or not. In the first case F is nested in F'' and F'' is nested in F'.

Consider now the third nesting repetition Q_3^1 of sequence $(H_1, H_2, \ldots, H_x)$ (see Figs. 8.20(a) and 8.20(b)). It is easy to see that if Q_3^1 is nested between Q_1^1 and Q_2^1, then there exists a nesting of depth 1 because Q_3^1 contains a defect at a different channel. So we have to consider the cases when the repetitions create the nesting by strictly going either outward or inward. By this we mean that the i-th repetition Q_i^1 has to be placed such that either Q_i^1 is nested inside Q_{i-1}^1 (inward) or vice versa (outward). Without loss of generality, we assume the latter (see Fig. 8.20(c)).

Consider now a defect in a channel c, with $1 < c < k$, at a certain repetition Q_i^h. Since the path is moving outward, the connection between channels $c-1$ and $c+1$ blocks visibility for the following repetitions to the part of the channel segment where vertices of channel c were placed till that repetition (see Fig. 8.21(a) for an example with $c = 3$).

A possible placement for the vertices of c in the following repetitions that does not increase the depth of the nesting could be in the same part of the channel segment where vertices of a channel c', with $c' \neq c$, were placed till that repetition. We call *shift* such a move. However, in order to place vertices of c and of c' in the same zone, all the vertices of c belonging to the current cell have to be placed there (see dashed lines in Fig. 8.21(b), where $c' = c+1$), which implies that a further defect in channel c at one of the following repetitions encloses all the vertices of each of the previously drawn cells, hence separating them with a straight line from the following cells. Hence, also the vertices of c' have to perform a shift to a channel c'', with $c \neq c'' \neq c'$. Again, if the vertices of c' and of c'' lie in the same zone, we have two cells that are separated by a straight line and hence also the vertices of c'' have to perform a shift. By repeating such an argument we conclude that the only possibility for not having vertices of different channels lying in the same zone is to shift all the channels $c, \ldots, x$ and to go back to channel 1 for starting the following repetition in a completely different region (see Fig. 8.22, where the following repetition is performed completely below the previous one). However, this implies that there exist two repetitions in one configuration that have to be separated by a straight line and therefore are independent, in contradiction to our assumption. Therefore, we can assume that, after $3 \cdot x + 1$ repetitions, we arrive at a nesting of depth 1. By repeating this argument we arrive after $3 \cdot x \cdot 6$ repetitions at the nesting of depth 6 claimed in the lemma. □

Lemma 8.9. *If an extended formation lies in a part of the channel that contains only 1-side connections, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Proof: First observe that, by Lemma 8.8, there exists a k-nesting with $k \geq 6$ in any extended formation EF.

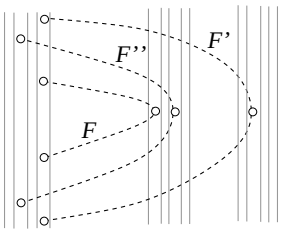


Figure 8.24: Illustration for the case with only 1-side connections.

Consider two nested formations $F, F' \in EF$ belonging to the k-nesting. Such formations, by definition, belong to the same channel. Consider now the formation $F'' \in EF$ belonging to a different channel such that F is nested in F'' and F'' is nested in F'. Since each pair of channel segments have a 1-side connection, we have that F'' blocks visibility for F' on the channel segment used by F for the nesting (see Fig. 8.24). Hence, F' has to use a different channel segment to perform its nesting, which increases by one the number of used channel segments for each level of nesting. Since the tree supports at most 4 channel segments, the statement follows. □

Lemma 8.10. *Consider two channels ch_p, ch_q with the same intersections. Then, none of channels ch_i, where $p < i < q$, have an intersection that is disjoint with the intersections of ch_p and of ch_q.*

Proof: The statement follows from the fact that the channel borders of ch_p and ch_q delimit the channel for all joints between p and q. So, if any channel ch_i, with $p < i < q$, had an intersection different from the one of ch_p and ch_q, it would either intersect with one of the channel borders of ch_p or ch_q or it would have to bend around one of the channel borders, hence crossing a straight line twice. □

Lemma 8.11. *Consider an x-nesting of a sequence of extended formations on an intersection $I_{(a,b)}$, with $a \leq 2$. Then, there exists a triangle t in the nesting that separates some of the triangles nesting with t from the bending area $b(a, a+1)$ (or $b(a-1, a)$).*

Proof: Consider three extended formations $EF_1(H_1), EF_2(H_1), EF_3(H_1)$ lying in a channel ch_1 and two extended formations $EF_1(H_2), EF_2(H_2)$ lying in a channel ch_2 such that all the channels of the sequence of extended formations are between ch_1 and ch_2 and there is no formation $F \notin EF(H_1), EF(H_2)$ nesting between $EF_1(H_1), EF_2(H_1), EF_3(H_1)$ and $EF_1(H_2), EF_2(H_2)$. Suppose, without loss of generality, that the bending point of ch_1 is enclosed into the bending point of ch_2.

Consider a formation $F_1 \in EF_1(H_1)$ nesting with a formation $F_1' \in EF_1(H_2)$. We have that the connections from F_1' to channel segment a and back has to go around the vertex placed by F_1 on channel segment a. Therefore, at least one of the connections of F_1' cuts all the channels between ch_1 and ch_2, that is, all the channels where the sequence of extended formations is placed. Such a connection separates the vertices of F_1 from the vertices of a formation $F_2 \in EF_2(H_1)$ on channel segment a. Therefore, at least one of the connections of F_2 to channel segment a cuts either all the channels in channel segment a or all the channels in channel segment $a+1$ (or $a-1$), hence becoming a blocking cut

for such channels. It follows that all the formations nesting inside F_2 on such channels can not place vertices in the bending area $b(a, a+1)$ (or $b(a-1, a)$) outside F_2. □

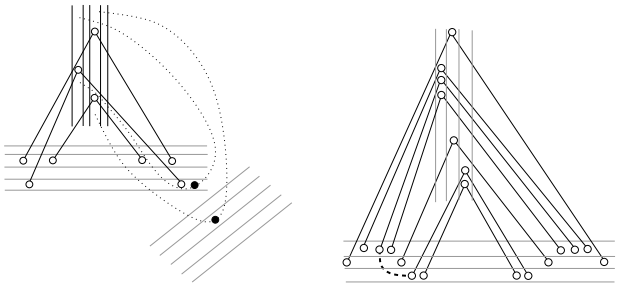
Lemma 8.12. *In a situation as described in Proposition 8.2, not all the extended formations in a sequence of extended formations can place turning vertices in the same channel segment.*

Proof: Assume, for a contradiction, that all the turning vertices are in the same channel segment. Consider a sequence of extended formations SEF and the extended formations in SEF using one of the sets of channels $\{H_1, \ldots, H_4\}$.

We first show that in SEF there exist some extended formations using connections in α configuration and some using connections in β configuration on channels $\{H_1, \ldots, H_4\}$. Consider the continuous subsequence of extended formations $EF(H_1), \ldots, EF(H_3)$ in SEF. Assume that all the turning vertices of these extended formations are in α configuration. Consider a further subsequence of SEF on the same set of channels with a defect at H_2. Then, the connection between H_1 and H_3 crosses H_2, thereby blocking any further $EF(H_2)$ from being in α configuration. Hence, when considering another subsequence of SEF on the same set of channels which does not contain defects at $H_1, \ldots, H_3$, either the extended formation $EF(H_2)$ is in β configuration or it uses another channel segment to place the turning vertex.

So, consider two channels H_1, H_2 such that there exists an extended formation $EF(H_1)$ in α configuration and an extended formation $EF(H_2)$ in β configuration. Since all the extended formations contain a triangle open on one side that is nested with triangle t, we consider five of such triangles, one for each set of channels H_2, H_3, H_4 and two for set H_1, such that four of the considered extended formations $EF(H_1), \ldots, EF(H_4)$ are continuous in SEF and the other one $EF'(H_1)$ is the first extended formation on the set of channels H_1 following $EF(H_4)$ in SEF.

Note that, if a triangle of an extended formation $EF(H_k)$ is nested in a triangle of an extended formation $EF(H_s)$ and the triangle of $EF(H_s)$ is nested in a triangle of an extended formation $EF'(H_k)$, with $k < s$, then $EF(H_k)$ has to use a different channel segment to place its turning vertex (see Fig. 8.25(a)). Hence, the triangles have to be ordered according to the order of the used channels. Also, if the continuous path connecting two triangles $t_1 = (u, v, w), t_2 = (u', v', w')$ of consecutive extended formations $EF(H_s), EF(H_{s+1})$ connects vertex u to vertex w' (or u' to w) via the outer area, then a triangle of $EF(H_1)$ that occurs prior to $EF(H_s)$ and a triangle of $EF'(H_1)$ that occurs after $EF(H_{s+1})$ are nested with the triangle given by the connection of t_1 and



(a) (b)

Figure 8.25: (a) Two triangles from the same channel have to use different channel segments if a triangle of another channel is between them. Turning vertices are represented by black circles. (b) When a defect at H_2 in encountered, the connection between $EF(H_1)$ and $EF(H_3)$ does not permit the following $EF(H_2)$ to respect the ordering of triangles.

t_2 in an ordering different from the order of the channels.

Consider now the following subsequence of SEF having a defect at H_2. The connection of $EF(H_1)$ to $EF(H_3)$ in this subsequence blocks access for the following $EF(H_2)$ to the area where it would have to place vertices in order to respect the ordering of triangles (see Fig. 8.25(b)). Therefore, after 3 full repetitions of the sequence in SEF, at least one extended formation has to use a different channel segment to place its turning vertex. □

Lemma 8.13. *In a situation as described in Proposition 8.2, $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Proof: Consider two extended formations $EF(H_x), EF(H_1)$ that are consecutive in SEF. First note that the connection between $EF(H_x)$ and $EF(H_1)$ cuts all channels $2, \ldots, x-1$ in either channel segment cs_1 or cs_2. Since both of these extended formations are also connected to the bending area between channel segments cs_3 and cs_4, it is not possible for an extended formation $EF(s)$, with $s \in \{2, \ldots, x-1\}$, to connect from vertices above the connection between $EF(H_x)$ and $EF(H_1)$ to vertices below it by following a path to the bending area. Note, further, that if all the extended formations $EF(s)$, with $s \in \{2, \ldots, x-1\}$, are in the channel segment that is not cut by the connection between $EF(1)$ and $EF(x)$, then a connection is needed from cs_1 to cs_2 in channel x. However, by Lemma 8.12, after three defects in the subsequence of

$\{2, \ldots, x-1\}$ it is no longer possible for some extended formation $EF(s)$, with $s \in \{2, \ldots, x-1\}$, to place its turning vertex in the same channel segment. Therefore, different channel segments have to be used by the extended formation $EF(s)$, with $s \in \{2, \ldots, x-1\}$. However, since the path is continuous and since the connection between $EF(H_x)$ and $EF(H_1)$ is repeated after a certain number of steps, we can follow that the path creates a spiral. Also, we note that, in order to respect the order of the sequence, it will be impossible for the path to reverse the direction of the spiral. Hence, once a direction of the spiral has been chosen, either inward or outward, all the connections in the remaining part of the sequence have to follow the same. This implies that, if a connection between $EF(s)$ and $EF(s+1)$ changes channel segment, that is, it is performed in a different channel segment than the one between $EF(s-1)$ and $EF(s)$, then all the connections of this type have to change. However, when a defect at channel $s+1$ is encountered, also the connection between $EF(s)$ and $EF(s+2)$ has to change channel segment, thereby making impossible for any future connection between $EF(s)$ to $EF(s+1)$ to change channel segment. Therefore, after a whole repetition of the sequence of SEF containing defects at each channel, all the extended formations have to place their turning vertices in the same channel segment, which is not possible, by Lemma 8.12, hence proving the statement □

Lemma 8.14. *If a shape contains an intersection $I_{(1,3)}$ and does not contain any other intersection that is disjoint with $I_{(1,3)}$, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Proof: First observe that only the intersections $I_{(2,4)}$ and $I_{(1,4)}$ are not disjoint with $I_{(1,3)}$ and could occur at the same time as $I_{(1,3)}$. By Lemma 8.8, there exists at least a nesting greater than, or equal to, 6. Each of such nestings has to take place either at intersections $I_{(1,3)}$, $I_{(2,4)}$ or at $I_{(1,4)}$. Remind that, by Property 8.4, 1-vertices can only be placed in cs_1 or cs_2. Also, the sorting of head vertices to avoid a region-level nonplanar trees can only be done by placing vertices into cs_3 or cs_4. This implies that the stabilizers have to be placed in cs_1 or cs_2. Note that the stabilizers also work as 1-vertices in the tails of other cells. This means that if there exist seven sets of tails that can be separated by straight lines, then there exist a region-level nonplanar tree, by Lemma 8.6. Observe that, by nesting them according to the sequence, the previous condition would be fulfilled. This means that we have either a sorting or other nestings. We first show that there exist at most two x-nestings with $x \geq 6$. Every x-nesting has to take place at either $I_{(1,3)}$, $I_{(2,4)}$ or $I_{(1,4)}$. We assume, w.l.o.g., to have to deal with the greatest possible number of

intersections.

Consider the case $I^h_{(2,4)}$ (see Fig 8.26(a)). Observe that intersections $I_{(1,4)}$ and $I_{(1,3)}$ are either both high or both low and use channel segment cs_1. Also, every connection from cs_1 to cs_4 cuts either cs_2 or cs_3 and, if one of these connections cuts cs_2, then every nesting cutting cs_1 closer to $b(1,2)$ has to cut cs_2. Hence, we can consider all the connections to cs_4 as connections to cs_2 or cs_3. Also, since any connection cutting a channel segment is more restrictive than a connection inside the same channel segment, such two nestings can be considered as one. Finally, since such a nesting connects to $b_{(2,3)}$, it is not possible to have at the same time a nesting taking place at $I^h_{(2,4)}$. Hence, we conclude that only one nesting is possible in this case.

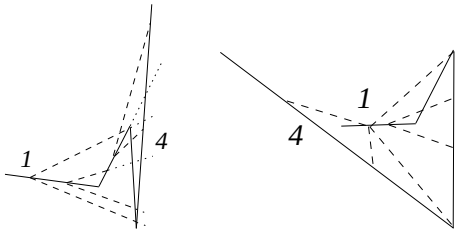


Figure 8.26: (a) Case $I_{(1,3)}$ $I^h_{(2,4)}$. (b) Case $I_{(1,3)}$ $I^l_{(2,4)}$.

Consider the case $I^l_{(2,4)}$ (see Fig 8.26(b)). Observe that 1-vertices can be placed at most in cs_2 and 2-vertices can be placed at most in cs_3. This means that the extended formations in every nesting have to visit these vertices. Therefore, if there exists both a nesting at $I_{(1,3)}$ and at $I_{(1,4)}$, then the connections to the 1- and 2-vertices in the bending areas $b(2,3)$ and $b(3,4)$ are such that every EF nesting at $I_{(1,4)}$ makes a nesting with the extended formations nesting at $I_{(1,3)}$. Hence, also in this case only one nesting is possible.

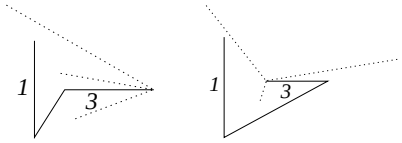
So we consider the unique nesting of depth $x \leq 6$ and we show that any way of sorting the nesting formations in the channels will cause separated cells, hence proving the existence of a nonplanar region-level tree. Consider four consecutive repetitions of the sequence of formations. It is clear that these formations are visiting areas of cs_1 and are separated by previously placed formations from other formations on the same channels. This will result in some cells to become separated in cs_1. Since, by Property 8.2, the number

of monotonically separated cells in cs_1 cannot be larger than 3, for any set of four such separated formations there exists a pair of formations F_1, F_2 that change their order in cs_1. These connections have to be made on either side of the nesting. If between this pair of formations there is a formation of a different channel, then this formation has to choose the other side to reorder with a formation outside F_1, F_2. We further note that, if there are two such connections F_1, F_4 and F_2, F_3 on the same side that are connecting formations of one channel, nested in the order F_1, F_2, F_3, F_4, and another connection on the same side between F_1', F_2' such that F_1' is nested between F_1, F_2 and F_2' between F_3, F_4, then this creates a 1-nesting. In the following we show that a nesting of depth at least 6 is reached.

Assume the repetitions of formations in the extended formation to be placed in the order a, b, c, d, e. If this order is not coherent with the order in which the channels appear in the sequence of formations inside the EF, then we have already some connections that are closing either side of the nesting for some formations. So we assume them to be in the order given by the sequence. Then, consider a repetition of formations with a defect at some channel C_i. We have that there exists a connection closing off at one side all the previously placed formations of C_i. However, there are sequences with defects also at channels C_{i+1} and C_{i-1}, which can not be realized on the same side as the defects at C_i. We generalize this to the fact that all the defects at odd channels are to one side, while the defects at even channels are to the other side. Since the path is continuous and has to reach from the last formation in a sequence again to the first one, the continuation of the path can only use either the odd or the even defects. This implies that, when considering three further repetitions of formations, the first and the third having a defect at a channel C_i and the second having no defect at C_i, there will be a nesting of depth one between these three formations. Since, by Lemma 8.9, there cannot be a nesting of depth greater than 5 at this place, we conclude that after 6 repetitions of such a triple of formations there will be at least two formations that are separated from each other. By repeating this argument we arrive after $7 \cdot 6 \cdot 2$ repetitions at either the existence of 7 formations that are separated on cs_1 and cs_2 or at the existence of a nesting of depth 6, both of which will not be drawable without the aid of another intersection that is able to support the second nesting of depth greater than 5. □

Lemma 8.15. *If there exists a sequence of extended formation in any shape containing an intersection $I_{(3,1)}$, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Proof: Consider a sequence of extended formation in a shape containing an intersection $I_{(3,1)}$. We show that $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding. Observe that there exist several possibilities for channel segment cs_4 to be placed. Either there exists no intersection of an elongation of one channel segment with another channel segment or there exists at least one of the intersections $I_{(1,4)}$, $I_{(4,2)}$, $I_{(4,1)}$ or $I_{(2,4)}$. If there are more than one of such intersections, then it is possible to have several nestings of depth x. We note that, if there exists the intersection $I_{(3,1)}$, then at least one of cs_1, cs_2, and cs_4 are part of the convex hull (see Fig. 8.27).



(a) (b)

Figure 8.27: If channel segment four is not part of the convex hull then either cs_1 or cs_2 is part of the convex hull. (a) Case $I^l_{(1,3)}$. (b) Case $I^h_{(1,3)}$.

First, we show that there exists a nesting at $I_{(3,1)}$.

Consider case $I^h_{(3,1)}$. We have that cs_2 is on the convex hull restricted to the first three channel segments and cs_4 can force at most one of cs_2 or cs_1 out of the convex hull. Hence, one of them is part of the convex hull. We distinguish the two cases.

Suppose cs_2 to be part of the convex hull. Assume there exists a nesting at $I_{(1,4)}$. From cs_4 the only possible connection without a 1-side connection is the one to cs_2, which, however, is on the convex hull. Hence, an argument analogous to the one used in Lemma 8.14 proves that the nesting at $I_{(2,4)}$ has size smaller than $7 * 12$, which implies that the rest of the nesting has to take place at $I_{(3,1)}$.

Suppose cs_1 to be part of the convex hull. Assume that there exists a nesting at $I_{(2,4)}$. Every connection from cs_4 has to be either to cs_1 or to cs_2, by Property 8.4. Since cs_2 is already part of the nesting, we have connections to cs_1. However, cs_1 is on the convex hull, hence allowing only 1-side connections. Therefore, an argument analogous to the one used in Lemma 8.14 proves that the nesting at $I_{(2,4)}$ has size smaller than $7 * 12$, which implies the rest of the nesting has to take place at $I_{(3,1)}$.

Consider case $I^l_{(3,1)}$. Since cs_2 is not part of the convex hull, either cs_1 or cs_4 are. If cs_1 is on the convex hull, then the same argument as before holds, while if cs_4 is on the convex hull, then no reordering is possible.

Clearly, if there is no intersection other than $I_{(3,1)}$, a nesting in the intersection $I_{(3,1)}$ has to be performed.

Hence, we conclude that a nesting with a depth of $7 * 12$ in every extended formation has to take place at $I_{(3,1)}$ (or at $I_{(4,1)}$, which can be considered as the same case).

By Lemma 8.11, the nesting in the bending area is limited. Every extended formation EF which has at least one vertex either in cs_3 or in cs_4 has a vertex in the bending area. Consider a sequence of extended formations SEF which uses only channels in this particular shape. It's obvious that all of these EF in SEF have to do a nesting at $I_{(3,4,1)}$. Observe that there exist two consecutive edges which are forming a triangle with cs_1, cs_2, and cs_3 by simply placing vertices inside the channel segments. Since every EF creates such triangles, there exists a triangle which is not in the bending area and such that there exists no other triangle between the bending area and this triangle. This triangle is separating the nesting area from the bending area in all but s extended formations. However, since every EF has to use both of such areas, the inner area of cs_3 (or cs_4) has to connect to the outer area of cs_3 (or cs_4). If cs_1 is on the convex hull, then there exist only 1-sided connections, which implies the statement, by Lemma 8.13. On the other hand, if cs_1 is not on the convex hull, then there exists $I_{(1,4)}$ and cs_4 can be also used to perform connections from the inner to the outer area. However, since cs_4 is on the convex hull, such connections are only 1-side. Hence, by Lemma 8.13, the statement follows. □

Lemma 8.16. *Let cs_i and cs_{i+1} be two consecutive channel segments. If there exists an ordered set $S := (1, 2, \ldots, 5)^3$ of extremal double cuts cutting cs_i and cs_{i+1} such that the order of the intersections of the double cuts with cs_i (with cs_{i+1}) is coherent with the order of S, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Proof: Suppose, for a contradiction, that such a set S exists. Assume first that cs_i and cs_{i+1} are such that the bendpoint of channel 5 encloses the bendpoint of all the other channels. Hence, any edge creating a double cut at a channel c has to cut all the channels c' with $c' > c$, either in cs_i or in cs_{i+1}. Refer to Fig. 8.28.

Consider the first repetition $(1, 2, \ldots, 5)$. Let e_1 be an edge creating a double cut at channel 1. Assume, without loss of generality, that e_1 cuts channel

segment cs_i. Observe that, for channel 1, the visibility constraints determined in channels $2, \ldots, 5$ in cs_i and in cs_{i+1} by the double cut created by e_1 do not depend on whether it is simple or non-simple. Indeed, by Property 8.6, edge e_1 blocks visibility to $b(i, i+1)$ for the part of cs_i where edges creating double cuts at channels $2, \ldots, 5$ following e_1 in S have to place their end-vertices.

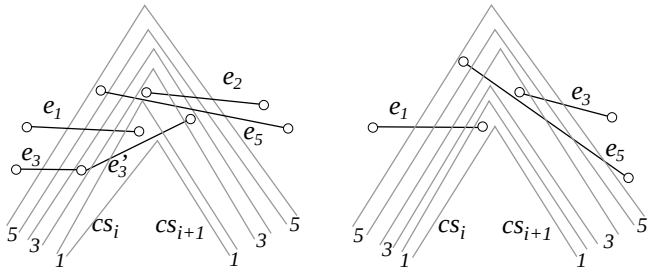


Figure 8.28: Proof of Lemma 8.16. (a) e_3 cuts cs_i. (b) e_3 cuts cs_{i+1}.

Then, consider an edge e_3 creating a double cut at channel 3 in the first repetition of $(1, 2, \ldots, 5)$.

If e_3 cuts cs_i (see Fig. 8.28(a)), then it has to create either a non-simple double cut or a simple one. However, in the latter case, an edge e'_3 between cs_i and cs_{i+1} in channel 3, which creates a blocking cut in channel 2, is needed. Hence, in both cases, channel 2 is cut both in cs_i and in cs_{i+1}, either by e_3 or by e'_3. It follows that an edge e_2 creating a double cut at channel 2 in the second repetition of $(1, 2, \ldots, 5)$ has to cut cs_{i+1}, hence blocking visibility to $b(i, i+1)$ for the part of cs_{i+1} where edges creating double cuts at channels $3, \ldots, 5$ following it in S have to place their end-vertices, by Property 8.6. Further, consider an edge e_5 creating a double cut at channel 5 in the second repetition of $(1, 2, \ldots, 5)$. Since visibility to $b(i, i+1)$ is blocked by e_1 and e_3 in cs_i and by e_2 in cs_{i+1}, e_2 has to create a non-simple double cut (or a simple one plus a blocking cut), hence cutting channel 4 both in cs_i and in cs_{i+1}. It follows that, by Property 8.5, an edge e_4 creating a double cut at channel 4 in the third repetition of $(1, 2, \ldots, 5)$ can place its end-vertex neither in cs_i nor in cs_{i+1}.

If e_3 cuts cs_{i+1} (see Fig. 8.28(b)), then it has to create a simple double cut. Again, by Property 8.6, edge e_3 blocks visibility to $b(i, i+1)$ for the part of

cs_{i+1} where edges creating double cuts following e_3 in S have to place their end-vertices. Hence, an edge e_5 creating a double cut at channel 5 in the first repetition of $(1, 2, \ldots, 5)$ cannot create a simple double cut, since its visibility to $b(i, i+1)$ is blocked by e_1 in cs_i and by e_3 in cs_{i+1}. This implies that e_5 creates a non-simple double cut (or a simple one plus a blocking cut) at channel 5, cutting either cs_i or cs_{i+1}, hence cutting channel 4 both in cs_i and in cs_{i+1}. It follows that, by Property 8.5, an edge e_4 creating a double cut at channel 4 in the second repetition of $(1, 2, \ldots, 5)$ can place its end-vertex neither in cs_i nor in cs_{i+1}.

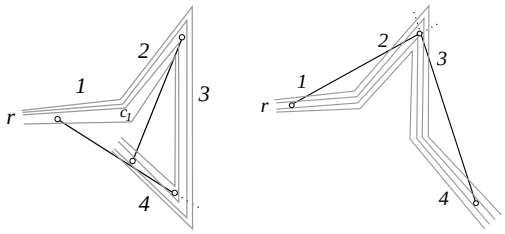
The case in which cs_i and cs_{i+1} are such that the bendpoint of 1 encloses the bendpoint of all the other channels can be proved analogously. Namely, the same argument holds with channel 5 playing the role of channel 1, channel 1 playing the role of channel 5, channel 3 having the same role as before, channel 4 playing the role of channel 2, and channel 2 playing the role of channel 4. Observe that, in order to obtain the needed ordering in this setting, 3 repetitions of $(1, 2, \ldots, 5)$ are needed. In fact, we consider channel 5 in the first repetition, channels 3 and 4 in the second one, and channels 1 and 2 in the third one. □

Lemma 8.17. *Each extended formation in shape $I^h_{(1,3)}$ $I^h_{(4,2)}$ creates double cuts in at least one bending area.*

Proof: Refer to Fig. 8.29(a). Assume, without loss of generality, that the first bendpoint of channel c_1 encloses the first bendpoint of all the other channels. This implies that the second and the third bendpoints of channel c_1 are enclosed by the second and the third bendpoints of all the other channels, respectively.

Suppose, for a contradiction, that there exists no double cut in $b(2, 3)$ and in $b(3, 4)$. Hence, any edge e connecting to $b(2, 3)$ (to $b(3, 4)$) is such that e and its elongation cut each channel once. Consider an edge connecting to $b(2, 3)$ in a channel c_i. Such an edge creates a triangle together with channel segments 3 and 4 of channel c_i which encloses the bending areas $b(3, 4)$ of all the the channels c_h with $h < i$ by cutting such channels twice. Hence, a connection to such a bending area in one of these channels has to be performed from outside the triangle. However, since in shape $I^h_{(1,3)}$ $I^h_{(4,2)}$ both the bending areas $b(2, 3)$ and $b(3, 4)$ are on the convex hull, this is only possible with a double cut, which contradicts the hypothesis. □

Lemma 8.18. *Every sequence of extending formations in shape $I^h_{(1,3)}$ $I^{h,l}_{(4,2)}$ contains an ordered set $(1, 2, \ldots, 5)^3$ of extremal double cuts with respect to bending area either $b(2, 3)$ or $b(3, 4)$.*



(a) (b)

Figure 8.29: (a) Shape $I^h_{(1,3)}\ I^h_{(4,\{1,2\})}$ has to connect at least one bend with double cuts. (b) Shape $I^h_{(1,3)}\ I^l_{(4,2)}$ has to connect bend $b(2,3)$ with double cuts.

Proof: Shape $I^h_{(1,3)}\ I^h_{(4,2)}$ is similar to shape $I^h_{(1,3)}\ I^h_{(4,1)}$, depicted in Fig. 8.29(a), with the only difference on the slope of channel segment 4, which is such that its elongation crosses channel segment 2 and not channel segment 1. Shape $I^h_{(1,3)}\ I^l_{(4,2)}$ is depicted in Fig. 8.29(b).

Assume, without loss of generality, that the first bendpoint of channel c_1 is enclosed by the first bendpoint of all the other channels. This implies that the second bendpoint of channel c_1 encloses the second bendpoint of all the other channels.

First observe that bending area $b(2,3)$ is on the convex hull, both in shape $I^h_{(1,3)}\ I^h_{(4,2)}$ and in shape $I^h_{(1,3)}\ I^l_{(4,2)}$.

Also, observe that all the extended formations have some vertices in $b(2,3)$ and in $b(3,4)$, and hence all the extended formations have to reach such vertices with path-edges.

In shape $I^h_{(1,3)}\ I^h_{(4,2)}$, by Lemma 8.17, there exist double cuts either in $b(2,3)$ or in $b(3,4)$, while in shape $I^h_{(1,3)}\ I^l_{(4,2)}$ there exist double cuts in $b(2,3)$, since the only possible connections to $b(2,3)$ are from channel segments 1 and 4, which are both creating double cuts (see Fig. 8.29(b)). Hence, we consider the extremal double cuts of each extended formation with respect to one of $b(2,3)$ or $b(3,4)$, say $b(2,3)$.

However, every repetition of extended formations inside a sequence of extended formations contains a double defect at some channel. We show, with an argument similar to the one used in Lemma 8.8, that the presence of such double defects determines an ordering $(1,2,\ldots,5)^3$ of extremal double cuts af-

ter a certain number of repetitions of extended formations inside a sequence of extended formations. Namely, consider a double defect at channel i in a certain repetition. The connection between channels $i-1$ and $i+2$ cannot be performed in the same area as the connection between channels $i-1$ and i and between channels i and $i+1$ was performed in the previous repetition. Hence, such a connection has to be performed either in the same area as the connection between channels $i+1$ and $i+2$ was performed (see Fig. 8.30(b)), or in channel segment 4 (this is only possible in shape $I^h_{(1,3)}$ $I^l_{(4,2)}$, see Fig. 8.30(c)). Observe that, going to channel segment 4 to make the connection, then to channel segment 1, and finally back to $b(2,3)$, hence creating a spiral, implies that the considered double cut is not extremal (see Fig. 8.30(d)). Therefore, the only possibility to consider when channel segment 4 is used is to make the connection between channels $i-1$ and $i+2$ there and then to come back to $b(2,3)$ with a double cut. Hence, independently on whether channel segment 4 is used or not, the connection between channels $i-1$ and $i+2$ blocks visibility for the following repetitions to the areas where the connections between some channels were performed in the previous repetition. This implies that the ordering $(1^n, 2^n, \ldots, 5^n)$ of extremal double cuts cannot be respected in the following repetitions. In fact, a partial order $(i, i+1, i+2)^2$ is obtained in a repetition of formations creating extremal double cuts at channels $1, \ldots, 5$. Also, when two different double defects having a channel in common are considered, the effect of such defects is combined. Namely, consider a double defect at channel 3 in a certain repetition. Consider two sets of extended formations creating double cuts in $b(2,3)$ at channels $1, \ldots, 5$, respectively. Observe that the extended formations in these two sets could be placed in such a way that the ordering of their extremal double cuts is $(1, 1, 2, 2, \ldots, 5, 5)$. The same holds for the following occurrences of extended formations creating double cuts in $b(2,3)$ at channels $1, \ldots, 5$, respectively. Clearly, in this way an ordering $(1^n, 2^n, \ldots, 5^n)$ could be achieved and hence an ordered set $(1, 2, \ldots, 5)^3$ of double cuts would be never obtained (see Fig. 8.30(a)). The connection between channels 2 and 5 blocks visibility to the areas where the connection between 2 and 3 and between 3 and 4 were performed at the previous repetitions (see Fig. 8.31(a)).

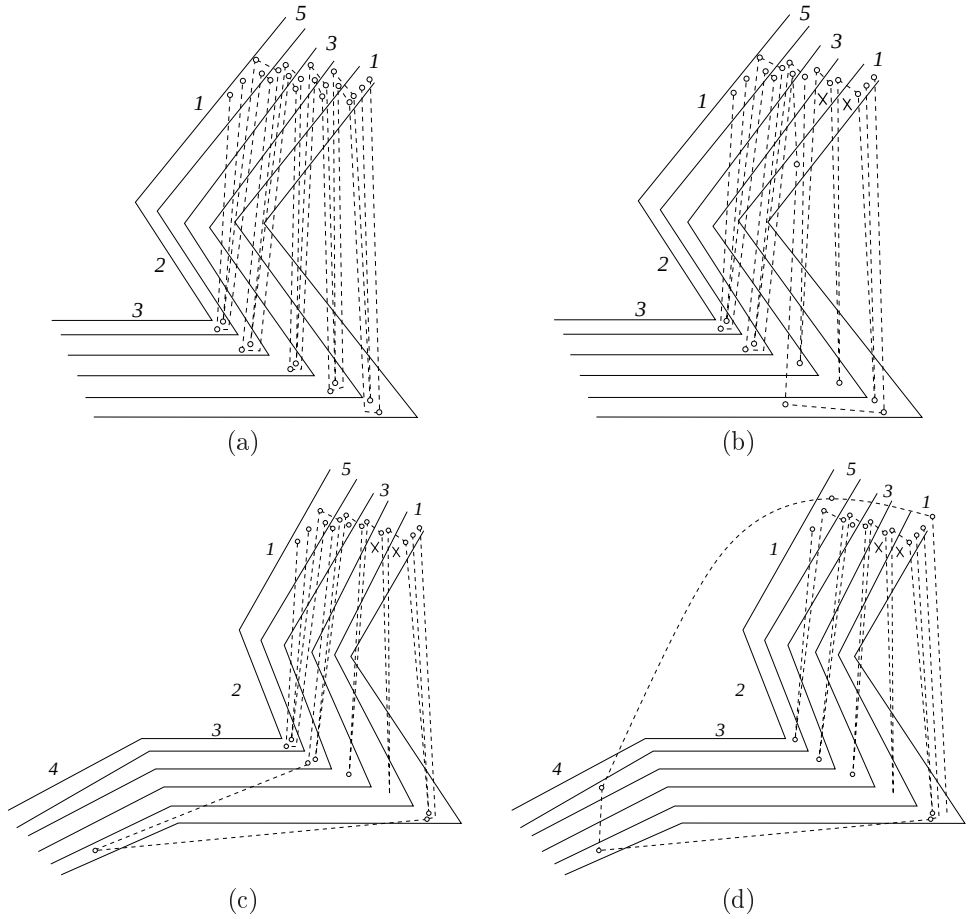


Figure 8.30: (a) The ordering of the extremal double cuts is $(1, 1, 2, 2, \ldots, 5, 5)$. (b) and (c) When a double defect is encountered, the connection between channels $i-1$ and $i+2$ cannot be performed in the same area as the connection between channels $i-1$ and i and between channels i and $i+1$ was performed in the previous repetition: (b) The connection is performed in the same area as the connection between channels $i+1$ and $i+2$ was performed. (c) The connection is performed in channel segment 4. (d) If channel segment four is used to spiral, the considered double cut was not extremal.

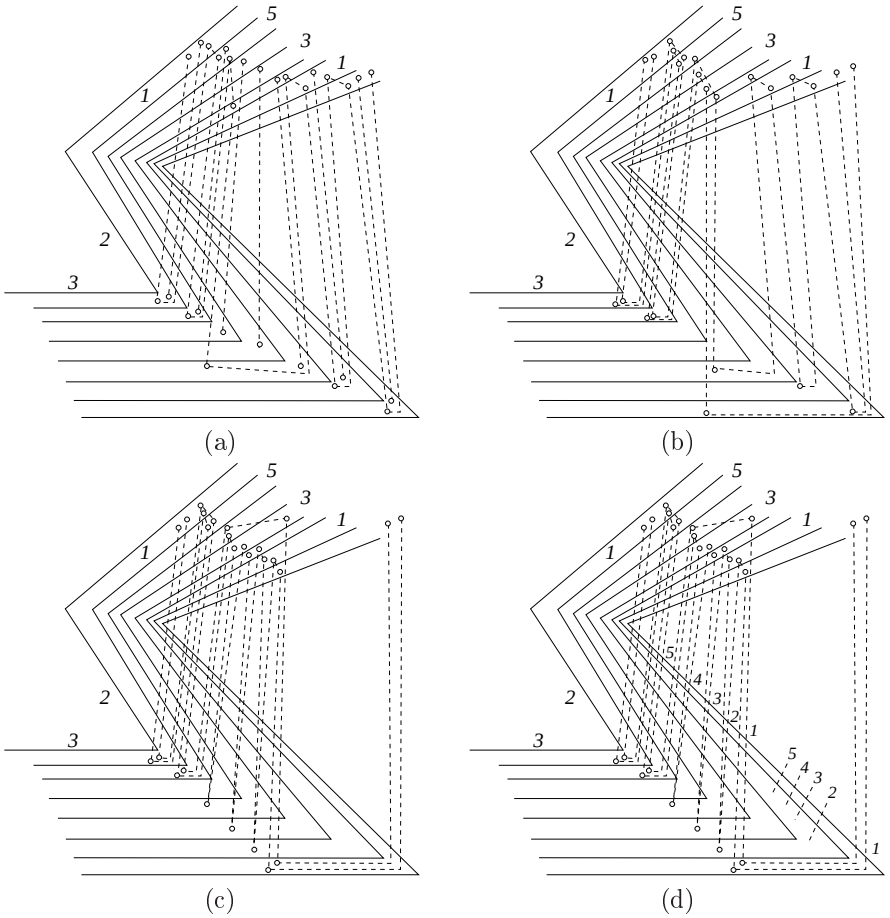


Figure 8.31: (a) A repetition with a double defect in channel 2 is considered. (b) A repetition with a double defect in channel 0 is considered. (c) A repetition without any double defect in channels $1, \ldots, 5$ is considered. (d) An ordered set $(1, \ldots, 5)$ is obtained.

Then, consider a double defect at channel 1 in a following repetition. We have that the connection between channels 0 and 3 can not be performed where the connection between 2 and 3 was performed in the previous repetitions, since such an area is blocked by the presence of the connection between channels 2 and 5. Hence, a double cut at channel 3 has to be placed after the double cut at channel 5 created in the previous repetition (see Fig. 8.31(b)). Consider now a further repetition with a defect not involving any of channels $1, \ldots, 5$. The region where the connection from 1 to 2 was performed in the previous repetitions is blocked by the connection between 0 and 3 and hence a double cut at channel 1 has to be placed after the one at channel 3 of the previous repetition, which, in its turn, was created after the one at channel 5 (see Fig. 8.31(c)). Also, all the double cuts at channels $2, \ldots, 5$ have to be placed after the double cut at 1, and hence a shift of the whole sequence $1, \ldots, 5$ after the double cut at 5 is performed and an ordered set $(1, 2, \ldots, 5)^2$ is obtained (see Fig. 8.31(d)). Observe that at most two sets of repetitions of extended formation inside a sequence of extended formations such that each set contains a double defect at each channel are needed to obtain such a shift. By repeating such an argument we obtain another shifting of the whole sequence $(1, \ldots, 5)$, which results in the desired ordered set $(1, 2, \ldots, 5)^3$. We have that a set of repetitions of extended formation containing a double defect at each channel is needed to obtain the first sequence $(1, 2, \ldots, 5)^2$, then two of such sets are needed to get to $(1, 2, \ldots, 5)^2$, and two more are needed to get to $(1, 2, \ldots, 5)^3$, which proves the statement.

Observe that, if it were possible to partition the defects into two sets such that there exists no pair of defects involving a common channel inside the same set, then such sets could be independently drawn inside two different areas and the effects of the defects could not be combined to obtain $(1, 2, \ldots, 5)^3$. However, since each double defect involves two consecutive channels, at least three sets are needed to obtain a partition with such a property. In that case, however, an ordered set $(1, 2, \ldots, 5)^3$ could be obtained by simply considering a repetition of $(1, 2, \ldots, 5)$ in each of the sets. □

Lemma 8.19. *If channel segment cs_2 is part of the convex hull, then $\mathcal{T}$ and $\mathcal{P}$ do not admit any geometric simultaneous embedding.*

Proof: First observe that, with an argument analogous to the one used in Lemma 8.14, it is possible to show that there exists a nesting at intersection $I_(4, 1, 2)$. Then, by Property 8.4, every vertex that is placed in cs_4 is connected to two vertices that are placed either in cs_1 or in cs_2. Hence, the continuous path connecting to a vertex placed in cs_4 creates a triangle, having one corner

in cs_4 and two corners either in cs_1 or in its elongation, which cuts cs_4 into two parts, the inner and the outer area.

By Lemma 8.11, not all of these triangles can be placed in the bending area $b(3,4)$. Hence, every extended formation, starting from the second of the sequence, have to place their vertices in both the inner and the outer area of the triangle created by the first one.

Observe that, in order to connect the inner to the outer area, the extended formations can only use 1-side connections. Namely, cs_1 creates a 1-side connection. Channel segment cs_2 is on the convex hull. Since, by Property 8.4, every vertex that is placed in cs_3 is connected to two vertices that are placed either in cs_1 or in cs_2, also cs_3 creates a 1-side connection.

From this we conclude that in this configuration the preconditions of Proposition 8.2 are satisfied, and hence the statement follows. □

8.6 An Algorithm for the Geometric Simultaneous Embedding of a Tree of Depth 2 and a Path

In this section we describe an algorithm for constructing a geometric simultaneous embedding of any tree $\mathcal{T}$ of depth 2 and any path $\mathcal{P}$. Refer to Fig. 8.32.

Start by drawing the root r of $\mathcal{T}$ on the origin in a coordinate system. Choose a ray R_1 emanating from the origin and entering the first quadrant, and a ray R_2 emanating from the origin and entering the fourth quadrant. Consider the wedge W delimited by R_1 and R_2 and containing the positive x-axis. Split W into t wedges $W_1, \ldots, W_t$, in this clockwise order around the origin, where t is the number of vertices adjacent to r in $\mathcal{T}$, by emanating $t-2$ equispaced rays from the origin.

Then, consider the two subpaths $\mathcal{P}_1$ and $\mathcal{P}_2$ of $\mathcal{P}$ starting at r. Assign an orientation to $\mathcal{P}_1$ and $\mathcal{P}_2$ such that the two edges $(r,u) \in \mathcal{P}_1$ and $(r,v) \in \mathcal{P}_2$ incident to r in $\mathcal{P}$ are exiting r.

Finally, consider the t subtrees $\mathcal{T}_1, \ldots, \mathcal{T}_t$ of $\mathcal{T}$ rooted at a node adjacent to r, such that $u \in \mathcal{T}_1$ and $v \in \mathcal{T}_t$.

The vertices of a subtree $\mathcal{T}_i$ are drawn inside wedge W_i, in such a way that:

- vertex u is the vertex with the lowest x-coordinate in the drawing, except for r;
- vertices belonging to $\mathcal{P}_1$ are placed in increasing order of x-coordinate according to the orientation of p_1;

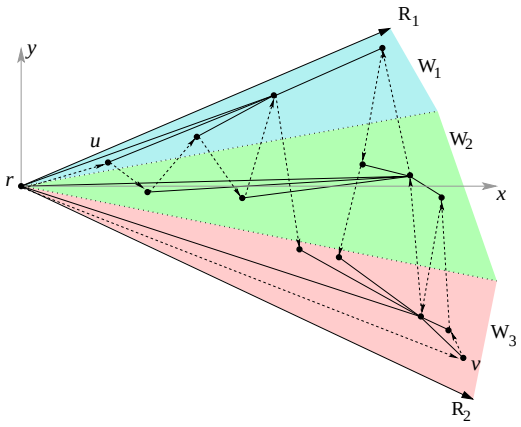


Figure 8.32: A tree with depth two and a path always admit a geometric simultaneous embedding.

- vertex v is the vertex with the highest x-coordinate in the drawing;
- vertices belonging to $\mathcal{P}_2 \setminus r$ are placed in decreasing order of x-coordinate according to the orientation of p_2, in such a way that the leftmost vertex of $\mathcal{P}_2 \setminus r$ is to the right of the rightmost vertex of $\mathcal{P}_1$; and
- no vertex is placed below segment $\overline{rv}$.

Since $\mathcal{T}$ has depth 2, each subtree $\mathcal{T}_i$, with $i = 1 \ldots, t$, is a star. Hence, it can be drawn inside its own wedge W_i without creating any intersection among tree-edges. Observe that the same holds even for subtree $\mathcal{T}_t$, where the wedge to consider is the part of W_t above segment $\overline{rv}$.

Since $\mathcal{P}_1$ and $\mathcal{P}_2 \setminus \{r\}$ are drawn in monotonic order of x-coordinate and are separated from each other, and edge (r, v) connecting such two paths is on the convex hull of the point-set, no intersection among path-edges is created.

From the discussion above, we have the following theorem.

Theorem 8.2 *A tree of depth* 2 *and a path always admit a geometric simultaneous embedding.*

8.7 Conclusions

In this chapter we have shown that there exist a tree $\mathcal{T}$ and a path $\mathcal{P}$ on the same set of vertices that do not admit any geometric simultaneous embedding, which means that there exists no set of points in the plane allowing a planar drawing of both $\mathcal{T}$ and $\mathcal{P}$. We obtained this result by extending the concept of level nonplanar trees [FK07b] to the one of region-level nonplanar trees. Namely, we showed that there exist trees that do not admit any planar embedding if the vertices are forced to lie inside particularly defined regions. Then, we constructed $\mathcal{T}$ and $\mathcal{P}$ so that $\mathcal{P}$ creates these particular regions and at least one of the many region-level nonplanar trees composing $\mathcal{T}$ has its vertices forced to lie inside them in the desired order. Observe that our result also implies that there exist two edge-disjoint trees that do not admit any geometric simultaneous embedding, which answers an open question posed in [GKV09], where the case of two non-edge-disjoint trees was solved.

It is important to note that, even if our counterexample consists of a huge number of vertices, it can also be considered as "simple", in the sense that the depth of the tree is just 4. In this direction, we proved that, if the tree has depth 2, then it admits a geometric simultaneous embedding with any path. This gives raise to an intriguing open question about whether a tree of depth 3 and a path always admit a geometric simultaneous embedding or not.

Part V

Clustered Graphs

Chapter 9

c-Planar Clustered Graphs

In this chapter we deal with clustered graphs and their drawings.

A *clustered graph* is a graph together with a hierarchy tree describing how vertices are grouped into clusters. Clustered graphs are widely used in applications where it is needed at the same time to represent relationships between entities and to group entities with semantic affinities. For example, consider the graph representing the Internet network, where the vertices are the routers and the edges are the links among them. In this case it is useful to group geographically close routers into areas, which in turn can be further grouped into Autonomous Systems. Also, the clustered model can be effectively applied to represent graphs at different levels of abstractions, showing semantic relations between vertices and making easy the navigation of large graphs.

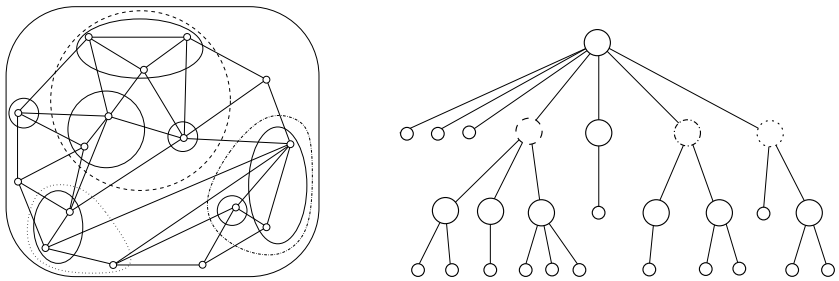
When considering the problem of drawing a clustered graph, not only the placement of the vertices and the drawing of the edges have to be decided, but also the drawing of the clusters, which are usually represented as closed regions enclosing all and only the vertices belonging to the cluster they represent. Then, the concept of planarity has to be extended to the one of *c-planarity* in order to deal with the representation of the clusters. It is interesting to observe that, while testing the planarity of a given graph is a linear-time solvable problem, testing whether a given clustered graph admits a c-planar drawing is still a problem of unknown complexity and represents one of the most studied problems in Graph Drawing in the last years.

In this chapter we introduce definitions about clustered graphs (Sect. 9.1) and their drawings (Sect. 9.2), and we present the state of the art about c-planarity testing (Sect. 9.3). Also, in Sect. 9.3, we describe a generalization of

the c-planarity testing problem, in which it is possible to split a cluster into two smaller clusters whenever a non-c-planarity of the input clustered graph is found. Then, the such a problem asks for the minimum number of splits to make the clustered graph c-planar.

9.1 Clustered Graphs

A *clustered graph* is a pair $C(G,T)$, where G is a graph, called the *underlying graph*, and T is a rooted tree, called the *inclusion tree*. The leaves of T are the vertices of G, while each internal node μ of T corresponds to the subset (called *cluster*) of the vertices of G that are leaves of the subtree of T rooted at μ. A clustered graph $C(G,T)$ with its inclusion tree T are depicted in Fig. 9.1. An edge (u,v) of G is *incident to a cluster μ of T* if u belongs to μ and v does not belong to μ. We denote by $\sigma(u_1,u_2,\ldots,u_k)$ the *smallest cluster* of T containing vertices $u_1,u_2,\ldots,u_k$ of G, i.e., the node of T containing all vertices $u_1,u_2,\ldots,u_k$ and such that none of its children in T, if any, contains all vertices $u_1,u_2,\ldots,u_k$. A cluster is *minimal* if it contains no cluster. A cluster μ is an *ancestor* (*descendant*) of a cluster ν if μ is an ancestor (descendant) of ν in T.



(a) (b)

Figure 9.1: (a) A clustered graph $C(G,T)$. (b) The inclusion tree T. Big circles represent clusters, while small circles represent vertices. Dashed, dotted, dash-dot alternated circles describe the correspondence of the nodes of T with the clusters. Such a correspondence is described only for the clusters that are children of the root.

A clustered graph $C(G,T)$ is *maximal* if G is a maximal planar graph. A clustered graph $C(G,T)$ is *internally-triangulated* if every internal face of G is delimited by a 3-cycle.

Families of Clustered Graphs

Besides the sub-families of clustered graphs that can be defined by simplifying the graph structure, that is, by restricting the underlying graph to belong to a simpler family of graphs, other interesting sub-classes can be defined by considering limitations in the cluster hierarchy. We say that a clustered graph $C(G,T)$ is *flat* if any path from the root to a leaf of T contains at most three nodes, that is, each cluster of T, except for the root cluster, is minimal. Another type of constraints that can be added in order to define simpler classes of clustered graphs is the one concerning the degree of connectivity of the subgraphs induced by the clusters. We say that a clustered graph $C(G,T)$ is *c-connected* if each cluster of T induces a connected subgraph of G, and *non-c-connected* otherwise. Further, we say that $C(G,T)$ is *completely connected* [CW06] if it is c-connected and the complement of each cluster of T induces a connected subgraph of G.

9.2 Drawing Clustered Graphs

A drawing Γ of a clustered graph $C(G,T)$ consists of a drawing of G and of a representation of each node μ of T as a simple closed region containing all and only the vertices belonging to μ. In this thesis, when we say "cluster", we refer both to the set of vertices belonging to the cluster and to the region representing the cluster in a drawing, the meaning of the word being clear from the context.

Consider an edge e of G and a node μ of T. If e crosses the boundary of the region representing μ in Γ more than once, we say that Γ contains an *edge-region crossing* between e and μ. Further, consider two nodes μ and ν of T. If the boundaries of the regions representing them in Γ have an intersection, we say that Γ contains an *region-region crossing* between μ and ν. We say that a drawing Γ of a clustered graph $C(G,T)$ is *c-planar* if G is drawn as a plane graph in Γ and if Γ contains no edge-region crossings and no region-region crossings. A clustered graph is c-planar if it admits a c-planar drawing.

A *c-planar embedding* of a c-planar clustered graph $C(G,T)$ is an equivalence class of c-planar drawings of C, where two c-planar drawings are equivalent if they have the same order of the edges incident to each vertex and the same order of the edges incident to each cluster.

The research on clustered graphs has been quite intense in the last years in the Graph Drawing community, mainly dealing with the following two problems:

1. Given a c-planar clustered graph C, construct a c-planar drawing of C respecting some aesthetic criteria.

2. Given a clustered graph C, test whether C admits a c-planar drawing.

We first give some results on the state of the art for the former problem.

Among the aesthetic criteria that are commonly requested to be respected for a drawing of a clustered graph (for a drawing of a graph, in general) to be readable, one of the most important is the one of having a small area. Concerning this subject, Eades *et al.* showed in [EFN99] how to construct $O(n^2)$-area c-planar orthogonal drawings and $O(n^2)$-area c-planar polyline drawings of c-planar clustered graphs with clusters drawn as axis-parallel rectangles. Also, Di Battista *et al.* [DDF07] showed algorithms and bounds for constructing small-area drawings of c-planar clustered trees within several drawing styles. In [HN09], the problem of constructing a c-planar drawing of a clustered graph $C(G,T)$ such that each face of G is represented as a convex polygon is studied.

When dealing with clustered drawings, another very important issue to consider is the one of providing a nice and readable geometric representation for the clusters, other than for the edges. In this area, the strongest result is perhaps the one that Eades *et al.* present in [EFLN06]. Namely, the authors show an algorithm for constructing c-planar straight-line drawings of c-planar clustered graphs in which each cluster is drawn as a convex region (see also a paper of Nagamochi and Kuroya [NK07]). Such an algorithm requires, in general, exponential area. However, in [FCE95a] Feng *et al.* have shown that such a bound is asymptotically optimal in the worst case. In Chapter 10 of this thesis we show an improvement of such a result by presenting an algorithm for constructing c-planar straight-line drawings of c-planar clustered graphs in which each cluster is drawn as an axis-parallel rectangle.

The state of the art on the c-planarity testing problem is presented in the following section.

9.3 Testing c-Planarity of Clustered Graphs

In this section we deal with the problem of testing c-planarity of a given clustered graph. Despite of the many research efforts spent in the last fifteen years on this problem, the question whether the c-planarity testing can be performed in polynomial time or not is still open, as its computational complexity is unknown.

However, many interesting results have been proved concerning the c-planarity testing of simpler families of clustered graphs obtained by adding constraints both on the graph structure and on the clustering structure.

A survey on the problem of testing the c-planarity of clustered graphs can be found in [CB05].

Concerning the possibility of considering simpler families of graphs in order to obtain efficient c-planarity testing algorithms, only few results have been obtained. In fact, a polynomial-time algorithm is known only for *cycles of clusters*, that is, flat clustered graphs in which the underlying graph is a simple cycle and the clusters are arranged in a cycle [CDPP05a]. Such a result was then improved to work for *clustered cycles*, that is, flat clustered graphs in which the underlying graph is a simple cycle and the clusters are arranged into an embedded plane graph [CDPP05b]. If the embedding is fixed and the faces are small, that is, each face contains at most five vertices, a characterization and an efficient testing algorithm exist [BF07]. Further, it has been shown a polynomial-time algorithm for *k-rib Eulerian graphs*, that is, clustered graphs such that the underlying graph is Eulerian and can be obtained from a 3-connected graph on k vertices, for some constant k, by multiplying and subdividing some edges [JKK+07]. Finally, a polynomial-time test exists if the clusters have at most four incident edges [SJTV08].

On the other hand, from the simplification of the clustering structure point of view, especially concerning the connectivity of the graphs induced by the clusters, the families of clustered graphs for which the problem can be solved is a bit wider.

If the clustered graph is c-connected, a first polynomial-time testing algorithm has been presented in 1995 [FCE95b], then improved in 1998 to a linear-time testing algorithm [Dah98]. A characterization and a consequent linear-time testing algorithm have been shown in [CBF+08]. Notice that every maximal c-planar clustered graph is c-connected. Namely, Feng *et al.* proved in [FCE95b] that a clustered graph $C(G,T)$ is c-planar if and only if edges can be added to G so that the resulting clustered graph $C'(G',T)$ is c-planar and c-connected. As no edge can be added to a maximal c-planar clustered graph without losing c-planarity, every maximal c-planar clustered graph is c-connected.

In 2008, Jelinek *et al.* [JJKL08] proved that the c-connectivity constraint can be slightly relaxed while maintaining the possibility of performing efficient c-planarity testings. Namely, they showed that a polynomial-time testing algorithm exists for embedded clustered graphs such that each cluster induces a subgraph of G with at most two connected components. If the clustered graph

is *almost connected*, there exists a polynomial-time testing algorithm [GJL+02]. A clustered graph is almost connected if either all the nodes corresponding to non-connected clusters are on the same path in the cluster hierarchy, or for each non-connected cluster its parent and all its siblings are connected. Almost connected clustered graphs are a generalization of c-connected clustered graphs, so as the *extrovert* clustered graphs [GLS05], for which a $O(n^3)$ time algorithm is known. Extrovert clustered graphs are such that each connected component of the graph induced by each cluster μ is connected by an edge to the graph induced by the parent ν of μ in the cluster hierarchy and the graph induced by ν is connected. Finally, if the clustered graph $C(G,T)$ is completely connected, it has been shown [CW06] that C is c-planar if and only if G is planar.

Generalization of the c-Planarity Testing

Let $C(G,T)$ be a clustered graph and suppose that a c-planar drawing of C is impossible or very difficult to find. A natural question is whether C admits a drawing where each cluster is represented by a small set of connected regions instead of a single connected region of the plane. In [AFP09] we formalize this concept by introducing the *split* operation, that replaces a cluster μ of T with two clusters μ_1 and μ_2 with the same parent as μ, and distributes the children of μ between μ_1 and μ_2. An example of split is depicted in Fig. 9.2. We search for the minimum number of splits turning C into a c-planar clustered graph. Formally, the corresponding decision problem is as follows:

Problem 9.1 Split-C-Planarity
Given a clustered graph $C = (G,T)$ and an integer $k \geq 0$, can $C(G,T)$ be turned into a c-planar clustered graph $C(G,T')$ by performing at most k split operations?

Split-C-Planarity is motivated not only by the practical need of drawing non-c-planar clustered graphs, but also by its implications on the c-planarity theory. In fact, the long-standing problem of testing c-planarity is a particular case of Split-C-Planarity, where zero splits are allowed. Therefore, Split-C-Planarity extends the c-planarity testing problem to a more general setting, where we are able to show the NP-hardness even for flat clustered graphs whose underlying graphs are paths or cycles.

Also, we show that Split-C-Planarity is NP-hard even for flat c-connected clustered graphs whose underlying graph is triconnected (hence even for flat

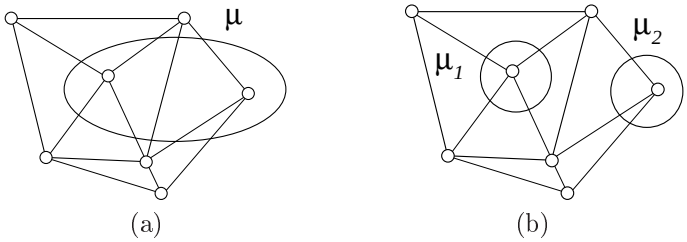


Figure 9.2: (a) A clustered graph $C(G,T)$ with a cluster μ containing two vertices. Observe that C is not c-planar, as μ contains two vertices that are separated by a cycle and it does not contain any vertex of such a cycle. (b) The clustered graph $C'(G,T')$ obtained by splitting μ into two clusters μ_1 and μ_2. Observe that C' is c-planar, as each cluster contains only one vertex.

c-connected embedded clustered graphs). On the other hand, we show that Split-C-Planarity is polynomial-time solvable for flat c-connected clustered graphs whose underlying graph is a biconnected series-parallel graph (both if the underlying graph has fixed or variable embedding) if the splits are assumed to preserve the c-connectivity of the graph.

Chapter 10

Straight-Line Rectangular Drawings of Clustered Graphs

In this chapter[1] we deal with a problem related to the research of "nice and readable" drawings of clustered graphs. Namely, we study a problem posed by Eades, Feng, Lin, and Nagamochi [EFLN06]:

Problem 10.1 *Does every c-planar clustered graph admit a straight-line c-planar drawing in which each cluster is represented by an axis-parallel rectangle?*

We answer this question in the affirmative by showing an algorithm to construct a *straight-line rectangular drawing* of every given c-planar clustered graph. This result improves a result of the same authors, stating that every c-planar clustered graph admits a straight-line c-planar drawing in which each cluster is represented by a convex polygon [EFLN06].

Since the construction used in this chapter is independent on the fact that the shape we consider is rectangular, the same property holds if clusters are represented by any convex shape fixed in advance. This result allows us to conclude that the c-planarity of clustered graphs is independent on the geometrical representation of the clusters. In some sense, the main theorem of this chapter can be considered as the analogous of Fary's Theorem [Far48], which states that the planarity of graphs is independent on the geometrical representation of the edges.

[1] Part of the work presented in this chapter is a joint work with Fabrizio Frati and Michael Kaufmann, appeared in [AFK09] and submitted to journal.

10.1 Introduction

Suppose that a c-planar embedded clustered graph C is given. How can the graph be drawn? Such a problem has been intensively studied in the literature and a number of papers have been presented for constructing c-planar drawings of c-planar clustered graphs within many drawing conventions.

Eades *et al.* show in [EFN99] how to construct $O(n^2)$-area c-planar orthogonal drawings and $O(n^2)$-area c-planar polyline drawings of c-planar clustered graphs with clusters drawn as axis-parallel rectangles. Di Battista *et al.* [DDF07] show algorithms and bounds for constructing small-area drawings of c-planar clustered trees within several drawing styles. The strongest result in the area is perhaps the one that Eades *et al.* present in [EFLN06]. Namely, the authors show an algorithm for constructing c-planar straight-line drawings of c-planar clustered graphs in which each cluster is drawn as a convex region (see also a paper of Nagamochi and Kuroya [NK07]). Such an algorithm requires, in general, exponential area. However, in [FCE95a] Feng *et al.* have shown that such a bound is asymptotically optimal in the worst case.

In this chapter we address a problem posed by Eades *et al.* in [EFL96, Fen97, EFLN06], namely whether every c-planar clustered graph has a *straight-line rectangular drawing*, i.e., a c-planar straight-line drawing in which each cluster is drawn as an axis-parallel rectangle. An example of a straight-line rectangular drawing of a c-planar clustered graph is given in Fig. 10.1. Eades *et al.* observe how pleasant and readable straight-line rectangular drawings are; however, they provide evidence that their algorithm [EFLN06] for constructing c-planar straight-line convex drawings of clustered graphs cannot be modified in order to get rectangular clusters without introducing edge-region crossings.

We show that every c-planar clustered graph has a straight-line rectangular drawing. Actually, we prove a stronger theorem stating that a straight-line rectangular drawing of a c-planar clustered graph exists for an arbitrary *convex-separated* drawing of its outer face, that is, a drawing satisfying some properties of convexity and of visibility among vertices and clusters.

Such a stronger result is proved by means of an inductive algorithm reminiscent of Fary's drawing algorithm for planar graphs [Far48]. Namely, the algorithm consists of three inductive cases. Each case considers a clustered graph C and performs an operation (removal of a cluster, split of the graph in correspondence of a separating 3-cycle, or contraction of an edge) turning C into a smaller clustered graph C' for which a straight-line rectangular drawing can be inductively constructed. Then, such a drawing can be easily augmented to a straight-line rectangular drawing of C.

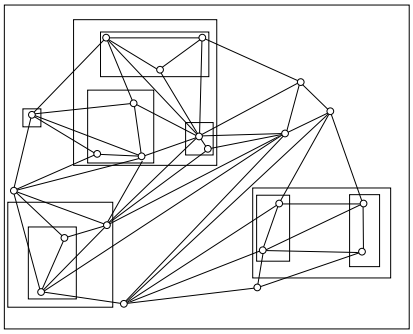


Figure 10.1: A straight-line rectangular drawing of a c-planar clustered graph.

When none of the three inductive cases applies, every cluster contains a vertex incident to the outer face. We use the term *outerclustered graph* for a clustered graph satisfying this property. We prove that every outerclustered graph admits a straight-line rectangular drawing even if a convex-separated drawing of its outer face is arbitrarily fixed, thus providing a base case for the above inductive algorithm for general clustered graphs. In order to draw an outerclustered graph C, we split it into three *linearly-ordered outerclustered graphs* (an even more restricted family of clustered graphs), we separately draw such graphs, and we compose the obtained drawings to get a drawing of C.

A linearly-ordered outerclustered graph is an outerclustered graph in which all the vertices of the underlying graph belong to a path in the inclusion tree. A drawing algorithm is provided for constructing a straight-line rectangular drawing of any linearly-ordered outerclustered graph $C(G,T)$ for an arbitrary convex-separated drawing of its outer face. Such an inductive algorithm finds a subgraph of G (a path plus an edge) that splits G into smaller linearly-ordered outerclustered graphs and draws such a subgraph so that the outer faces of the smaller linearly-ordered outerclustered graphs are convex-separated, thus allowing the induction to go through.

The rest of the chapter is organized as follows. In Sect. 10.2 we introduce some definitions about clustered graphs, linearly-ordered outerclustered graphs, and convex-separated drawings; in Sect. 10.3 we show an algorithm for linearly-ordered outerclustered graphs; in Sect. 10.4 we show an algorithm for outerclustered graphs; in Sect. 10.5 we show an algorithm for general clustered graphs; finally, in Sect. 10.6 we conclude and present some open problems.

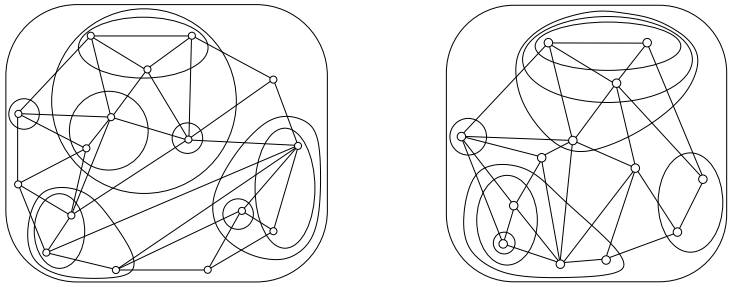
10.2 Preliminaries

In this chapter we are interested in *straight-line rectangular drawings* of clustered graphs, i.e., c-planar drawings such that each edge is represented by a straight-line segment and each cluster is represented by an axis-parallel rectangle. From now on, clustered graphs will be assumed to be c-planar, while drawings will be assumed to be straight-line rectangular.

In order to prove that every clustered graph admits a straight-line rectangular drawing, it suffices to prove that every maximal clustered graph admits a straight-line rectangular drawing, as every non-maximal c-planar clustered graph $C(G,T)$ can be augmented in linear time to a maximal c-planar clustered graph $C'(G',T)$ by adding dummy edges to G [JLP02]. We remark that every maximal c-planar clustered graph is c-connected.

From now on, we will always assume that the embedding (that is, the order of the edges incident to each vertex) and the outer face of any considered graph G is fixed in advance. We denote by $o(G)$ the outer face of G.

Fig. 10.2.a shows a biconnected internally-triangulated clustered graph.



(a) (b)

Figure 10.2: (a) A biconnected internally-triangulated clustered graph. (b) A biconnected internally-triangulated outerclustered graph.

Let $C(G,T)$ be a clustered graph. Let f be any face of G. Denote by $C_f(G_f,T_f)$ the clustered graph whose underlying graph G_f is the cycle induced by the vertices incident to f and whose inclusion tree T_f is the subtree of T induced by the clusters containing vertices incident to f. In particular, the *outer face of* $C(G,T)$ is the clustered graph $C_{o(G)}(G_{o(G)},T_{o(G)})$, that is simply denoted by C_o in the following. In Sect. 10.3, Sect. 10.4, and Sect. 10.5, we will prove that a drawing of a clustered graph can be constructed for an

arbitrary drawing of its outer face that satisfies some geometric properties to be described below. Then, we say that a straight-line rectangular drawing $\Gamma(C)$ of C *completes* a straight-line rectangular drawing $\Gamma(C_o)$ of C_o if the part of $\Gamma(C)$ representing C_o coincides with $\Gamma(C_o)$.

In order to prove that every clustered graph admits a straight-line rectangular drawing, it will be useful to study the class of clustered graphs that is defined below (see Fig. 10.2(b)).

Definition 10.1 *A c-planar clustered graph $C(G,T)$ is outerclustered if:*

- *O1: every cluster contains at least one vertex incident to $o(G)$;*
- *O2: the boundary of every cluster μ that does not contain all the vertices incident to $o(G)$ intersects $o(G)$ exactly twice, that is, it intersects exactly two edges $e_1(\mu)$ and $e_2(\mu)$ incident to $o(G)$; and*
- *O3: every edge (u,v) with $\sigma(u) = \sigma(v)$ is incident to $o(G)$.*

Let $C(G,T)$ be a biconnected internally-triangulated outerclustered graph and let $\mathcal{C}$ be any simple cycle in G such that the boundary of every cluster in T containing some but not all the vertices of $\mathcal{C}$ intersects $\mathcal{C}$ exactly twice. Denote by $C'(G',T')$ the clustered graph such that G' is the subgraph of G induced by the vertices incident to and internal to $\mathcal{C}$, and such that T' is the subtree of T induced by the clusters containing vertices of G'.

Lemma 10.1 *$C'(G',T')$ is a biconnected internally-triangulated outerclustered graph.*

Proof: Since G is biconnected and internally-triangulated, G' is biconnected and internally-triangulated, as well. We prove that C' satisfies Property $O1$ of Definition 10.1. Suppose that there exists a cluster μ in T' that does not contain any vertex incident to $o(G')$. Then, μ contains a vertex internal to G', otherwise it would not be a cluster in T'. Also, μ contains a vertex incident to $o(G)$, otherwise C would not be an outerclustered graph. Since the boundary of μ is a closed curve containing a vertex inside $\mathcal{C}$ and a vertex outside $\mathcal{C}$, then either μ contains a vertex of $\mathcal{C}$ or it intersects twice the same edge of $\mathcal{C}$, in both cases contradicting the c-planarity of C. Clustered graph C' satisfies Property $O2$ by hypothesis. We prove that C' satisfies Property $O3$. Suppose that there exists an edge (u',v') such that $\sigma(u') = \sigma(v')$ and u' is an internal vertex of G'. Then, u' is an internal vertex of G, as well, and C is not an outerclustered graph, a contradiction. □

An interesting subclass of the outerclustered graphs is considered in the following (see Fig. 10.3).

Definition 10.2 *A biconnected internally-triangulated outerclustered graph $C(G,T)$ is linearly-ordered if there exists a sequence $\mu_1, \mu_2, \ldots, \mu_k$ of clusters in T and an index $1 \leq h \leq k$, such that:*

- *LO1: for each vertex v_j of G, $\sigma(v_j) = \mu_i$, for some $1 \leq i \leq k$;*
- *LO2: let v_i and v_j be any two vertices incident to $o(G)$ such that $\sigma(v_i) = \mu_1$ and $\sigma(v_j) = \mu_k$; then $o(G)$ is delimited by two monotone paths $\mathcal{P}_1 = (v_i, v_{i+1}, \ldots, v_{j-1}, v_j)$ and $\mathcal{P}_2 = (v_i, v_{i-1}, \ldots, v_{j+1}, v_j)$, i.e., paths such that, if $\sigma(v_t) = \mu_a$ and $\sigma(v_{t+1}) = \mu_b$, then $a \leq b$ if $(v_t, v_{t+1}) \in \mathcal{P}_1$ and $b \leq a$ if $(v_t, v_{t+1}) \in \mathcal{P}_2$; and*
- *LO3: μ_{i+1} is the parent of μ_i, for each $1 \leq i < h$, and μ_{i+1} is a child of μ_i, for each $h \leq i < k$.*

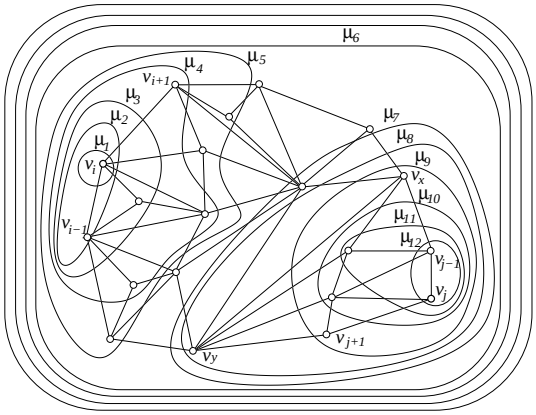


Figure 10.3: An outerclustered graph that is linearly-ordered according to the cluster sequence $\mu_1, \mu_2, \ldots, \mu_{12}$. Notice that $h = 6$.

Let $C(G,T)$ be a biconnected internally-triangulated outerclustered graph and suppose that C is linearly-ordered according to a sequence $\mu_1, \mu_2, \ldots, \mu_k$ of clusters of T. Let $v_i \in \mu_1$ and $v_j \in \mu_k$. Then, denote by V_1 (resp. by V_2) the vertex set containing all the internal vertices of $\mathcal{P}_1$ (resp. of $\mathcal{P}_2$).

In Sect. 10.3 we will prove that a drawing of any linearly-ordered outerclustered graph $C(G,T)$ can be obtained even if the drawing of C_o is arbitrarily fixed. However, we will deal with a constrained version of straight-line rectangular drawings that is defined below (see Fig. 10.4).

Definition 10.3 *A straight-line rectangular drawing $\Gamma(C_o)$ of C_o is a convex-separated drawing if:*

- *CS1: the polygon P representing $o(G)$ is convex;*
- *CS2: there exist two vertices v_i and v_j, with $\sigma(v_i) = \mu_1$ and $\sigma(v_j) = \mu_k$, such that the angle of P incident to v_i and the angle of P incident to v_j are strictly less than* 180°; *and*
- *CS3: for every pair of clusters μ and ν such that μ is the parent of ν in T and such that μ is not an ancestor of the smallest cluster containing all the vertices of $o(G)$, there exists a convex region $R(\mu,\nu)$ such that: (i) $R(\mu,\nu)$ is entirely contained inside $\mu \cap (P \cup int(P))$, where $int(P)$ denotes the interior of polygon P; (ii) for any cluster $\mu' \neq \mu$ and any child ν' of μ', $R(\mu,\nu)$ intersects neither $R(\mu',\nu')$ nor the boundary of μ'; (iii) $R(\mu,\nu) \cap P$ consists of two polygonal lines $l_1(\mu,\nu)$ and $l_2(\mu,\nu)$ such that $l_1(\mu,\nu)$ belongs to the polygonal line representing $\mathcal{P}_1$ in $\Gamma(C_o)$ and $l_2(\mu,\nu)$ belongs to the polygonal line representing $\mathcal{P}_2$ in $\Gamma(C_o)$; further, at least one endpoint of $l_1(\mu,\nu)$ (resp. of $l_2(\mu,\nu)$) lies on $e_1(\nu)$ (resp. on $e_2(\nu)$).*

Let $C(G,T)$ be an outerclustered graph with outer face $o(G)$ delimited by a cycle $\mathcal{C} = (v_i, v_{i+1}, \ldots, v_{j-1}, v_j, v_{j+1}, \ldots, v_{i-1}, v_i)$. Suppose that C is linearly-ordered according to a sequence $\Sigma = \mu_1, \mu_2, \ldots, \mu_k$ of clusters in T. Let (v_x, v_y) be a chord of $\mathcal{C}$. Consider the clustered graphs $C^1(G^1,T^1)$ and $C^2(G^2,T^2)$ such that G^1 (resp. G^2) is the subgraph of G induced by the vertices incident to and internal to cycle $\mathcal{C}^1 = (v_x, v_{x+1}, \ldots, v_{y-1}, v_y, v_x)$ (resp. to cycle $\mathcal{C}^2 = (v_y, v_{y+1}, \ldots, v_{x-1}, v_x, v_y)$), and such that T^1 (resp. T^2) is the subtree of T induced by the clusters containing vertices of G^1 (resp. of G^2).

Lemma 10.2 *$C^1(G^1,T^1)$ and $C^2(G^2,T^2)$ are linearly-ordered outerclustered graphs.*

Proof: We prove the statement for C^1, the proof for C^2 being analogous. Refer to Fig. 10.5.

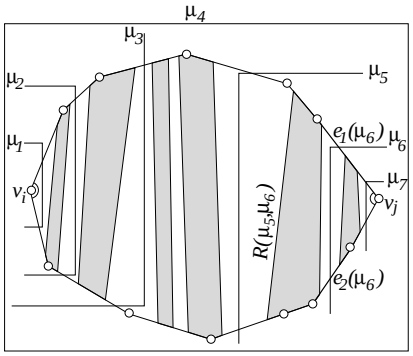


Figure 10.4: A convex-separated drawing of the outer face of a linearly-ordered outerclustered graph.

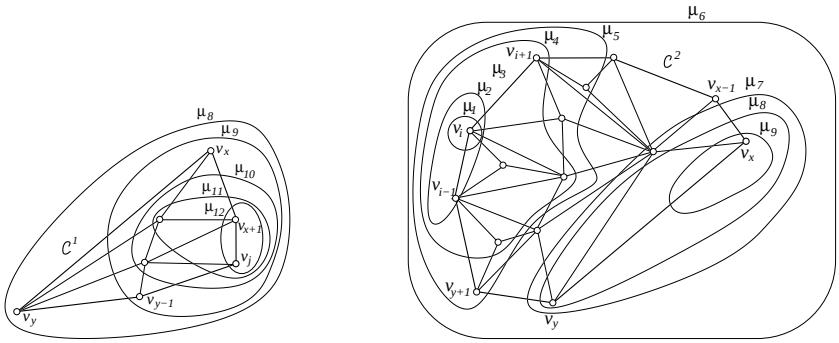


Figure 10.5: The linearly-ordered outerclustered graphs obtained by splitting the linearly-ordered outerclustered graph of Fig. 10.3 by chord (v_x, v_y). (a) $C^1(G^1, T^1)$. (b) $C^2(G^2, T^2)$.

We claim that C^1 is a biconnected internally-triangulated outerclustered graph. In order to prove the claim, by Lemma 10.1, it suffices to show that no cluster in T^1 exists that contains some but not all the vertices of C^1 and that does not intersect $\mathcal{C}^1$ exactly twice. Namely, the boundary of every cluster μ is a simple closed curve, and hence it intersects $\mathcal{C}^1$ an even number of times. Suppose that the boundary of μ does not intersect $\mathcal{C}^1$. Then, since μ contains

vertices of G^1, it contains all of such vertices. Suppose that μ intersects $\mathcal{C}^1$ at least four times. At most two of such intersections can be on the edges of $\mathcal{C}^1 \setminus (v_x, v_y)$, since C is an outerclustered graph. Then, the boundary of μ intersects (v_x, v_y) at least twice, contradicting the c-planarity of C.

Consider the subsequence Σ_1 of Σ induced by the clusters in T^1. Obtain a sequence Σ'_1 by removing from Σ_1 all the clusters that contain all the vertices of G^1 and that are different from $\sigma(v_x, v_{x+1}, \ldots, v_y)$, if any such a cluster exists. We claim that C^1 is linearly-ordered according to Σ'_1.

We prove that C^1 satisfies Property LO1 of Definition 10.2. By definition, Σ'_1 contains all the clusters of T^1 that contain vertices of G^1, except for each cluster μ that contains all the vertices of G^1 and that is different from $\sigma(v_x, v_{x+1}, \ldots, v_y)$. However, for any vertex v of G^1, $\sigma(v) \neq \mu$, because $\sigma(v_x, v_{x+1}, \ldots, v_y)$ is a descendant of μ and contains v.

We prove that C^1 satisfies Property LO2. Since C is linearly-ordered, $o(G)$ is delimited by two monotone paths $\mathcal{P}_1 = (v_i, v_{i+1}, \ldots, v_j)$ and $\mathcal{P}_2 = (v_i, v_{i-1}, \ldots, v_j)$. Suppose that (see Fig. 10.6(a)) both v_x and v_y belong to $\mathcal{P}_1$ and v_x precedes v_y in $\mathcal{P}_1$ (the other cases in which both v_x and v_y belong to $\mathcal{P}_1$ or both belong to $\mathcal{P}_2$ being analogous); then, the subpath of $\mathcal{P}_1$ between v_x and v_y and edge (v_x, v_y) are monotone paths delimiting $o(G^1)$; further, $\mathcal{P}_2$ and the path obtained from $\mathcal{P}_1$ by replacing the subpath between v_x and v_y with edge (v_x, v_y) are monotone paths delimiting $o(G^2)$. Suppose that v_x belongs to $\mathcal{P}_1$ and v_y belongs to $\mathcal{P}_2$. If $\sigma(v_x)$ precedes $\sigma(v_y)$ in Σ (see Fig. 10.6(b)), then the two monotone paths delimiting $o(G^1)$ are (i) the subpath of $\mathcal{P}_1$ between v_x and v_j and (ii) edge (v_x, v_y) plus the subpath of $\mathcal{P}_2$ between v_y and v_j; the two monotone paths delimiting $o(G^2)$ are (i) the subpath of $\mathcal{P}_1$ between v_i and v_x plus edge (v_x, v_y) and (ii) the subpath of $\mathcal{P}_2$ between v_i and v_y. If $\sigma(v_y)$ precedes $\sigma(v_x)$ in Σ, then the two monotone paths delimiting $o(G^1)$ are (i) edge (v_y, v_x) plus the subpath of $\mathcal{P}_1$ between v_x and v_j and (ii) the subpath of $\mathcal{P}_2$ between v_y and v_j; the two monotone paths delimiting $o(G^2)$ are (i) the subpath of $\mathcal{P}_1$ between v_i and v_x and (ii) the subpath of $\mathcal{P}_2$ between v_i and v_y plus edge (v_y, v_x).

We prove that C^1 satisfies Property LO3. Since C is linearly-ordered according to Σ and Σ'_1 is a subsequence of Σ, it suffices to show that any two consecutive clusters μ_x and μ_y of Σ'_1 are adjacent in T^1. Suppose, for a contradiction, that μ_y and μ_x are not adjacent in T^1. First, consider the case in which μ_x and μ_y are comparable, that is, μ_y is either an ancestor or a descendant of μ_x. If μ_y is an ancestor of μ_x (the case in which μ_x is an ancestor of μ_y being analogous), consider the parent μ_{x+1} of μ_x. Such a cluster contains all the vertices contained in μ_x, hence either μ_{x+1} belongs to Σ'_1, contradicting

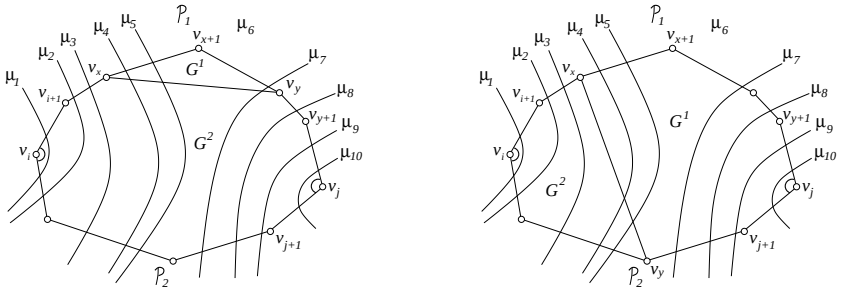


Figure 10.6: C^1 and C^2 satisfy LO2. (a) v_x and v_y belong to $\mathcal{P}_1$, and v_x precedes v_y in $\mathcal{P}_1$. (b) v_x belongs to $\mathcal{P}_1$, v_y belongs to $\mathcal{P}_2$, and $\sigma(v_x)$ precedes $\sigma(v_y)$ in Σ.

the fact that μ_x and μ_y are consecutive in Σ'_1, or μ_{x+1} contains all the vertices of G^1 and is different from $\sigma(v_x, v_{x+1}, \ldots, v_y)$. However, this would imply that also μ_y contains all the vertice s of G^1 and is different from $\sigma(v_x, v_{x+1}, \ldots, v_y)$, a contradiction to the fact that μ_y is in Σ'_1. Second, consider the case in which μ_x and μ_y are incomparable, that is, μ_y is neither an ancestor nor a descendant of μ_x. Again, consider the parent μ_{x+1} of μ_x. Such a cluster contains all the vertices contained in μ_x, still not being in Σ'_1. Hence, μ_{x+1} contains all the vertices of G^1 and is different from $\sigma(v_x, v_{x+1}, \ldots, v_y)$; this implies that $\mu_x = \sigma(v_x, v_{x+1}, \ldots, v_y)$, hence μ_y is a descendant of μ_x, a contradiction. □

Let $C(G,T)$, $C^1(G^1,T^1)$, and $C^2(G^2,T^2)$, Σ, v_x, and v_y be defined as above. Let Γ be any convex-separated drawing of C_o and let P be the polygon representing $o(G)$ in Γ; let v_i and v_j be any two vertices with $\sigma(v_i) = \mu_1$ and $\sigma(v_j) = \mu_k$ such that the angle of P incident to v_i and the angle of P incident to v_j are strictly less than 180° (such vertices exist by Property CS2 of Definition 10.3). Suppose that v_x and v_y are not collinear with any vertex of $o(G)$. Let Γ_1 and Γ_2 be the drawings of C^1_o and C^2_o obtained by drawing (v_x, v_y) in Γ as a straight-line segment. Denote by P^1 and P^2 the polygons representing $o(G^1)$ and $o(G^2)$ in Γ_1 and in Γ_2, respectively.

Lemma 10.3 *Γ_1 and Γ_2 are convex-separated drawings.*

Proof: We prove the statement for Γ_1, the proof for Γ_2 being analogous. Refer to Fig. 10.7. The drawing is straight-line and rectangular by construction.

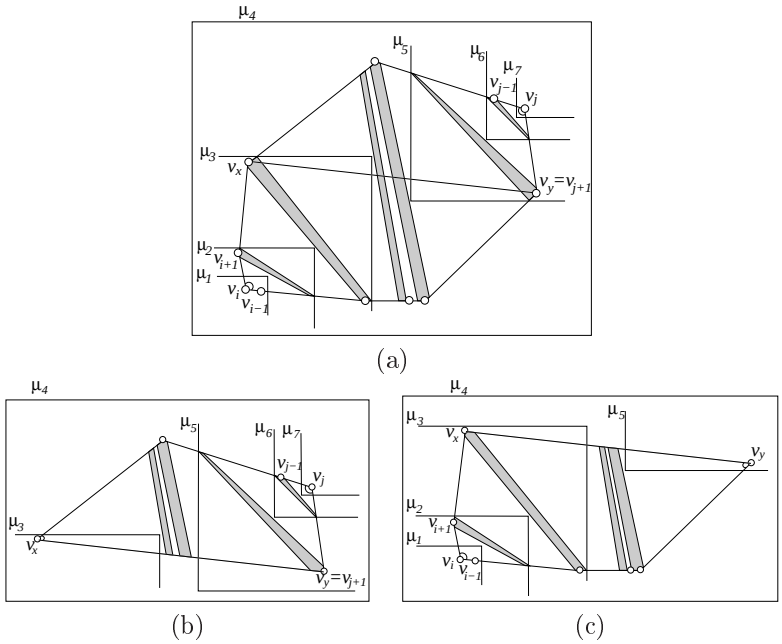


Figure 10.7: (a) A convex-separated drawing Γ of the outer face C_o of a linearly-ordered outerclustered graph C. (b) and (c) Convex-separated drawings Γ_1 and Γ_2 of the outer faces C_o^1 and C_o^2 of the linearly-ordered outerclustered graphs C^1 and C^2.

Since Γ is a convex-separated drawing, by Property CS1 of Definition 10.3, the polygon P representing $o(G)$ in Γ is convex. Further, by hypothesis, u_x and u_y are not collinear with any vertex of G. Hence, P^1 is convex, thus satisfying Property CS1.

Drawing Γ_1 has no region-region crossings, since each cluster is represented in Γ_1 by the same rectangle as in Γ. We prove that Γ_1 has no edge-region crossings. Refer to Fig. 10.8. Suppose that an edge-region crossing exists between an edge e and a cluster ν. Then, e is edge (v_x, v_y), otherwise Γ would not be c-planar. Cluster ν does not contain both of v_x and v_y otherwise, by the convexity of ν, e would be internal to ν; further, ν does not contain exactly one of v_x and v_y otherwise, by the convexity of ν, e would cross ν exactly once. It follows that ν contains neither v_x nor v_y. Consider the parent μ of ν in T_1.

Such a parent exists, otherwise ν would be the root of T, contradicting the fact that ν contains neither v_x nor v_y. Since Γ satisfies Property CS3, there exists a convex region $R(\mu, \nu)$ with the properties desc ribed in Definition 10.3 which "separates" ν from the rest of the drawing, thus avoiding an edge-region crossing between e and ν. More precisely, since Γ satisfies Property O2 of Definition 10.1, ν has exactly two incident edges $e_1(\nu)$ and $e_2(\nu)$ belonging to $o(G)$. Denote by $u(e_1(\nu))$ and $u(e_2(\nu))$ the endvertices of $e_1(\nu)$ and $e_2(\nu)$ belonging to ν. Denote by $p(l_1)$ the endpoint of $l_1(\mu, \nu)$ that lies on $e_1(\nu)$ (if both endpoints of $l_1(\mu, \nu)$ lie on $e_1(\nu)$, then $p(l_1)$ is the one that is closer to $u(e_1(\nu))$). Note that an endpoint of $l_1(\mu, \nu)$ lying on $e_1(\nu)$ exists as Γ satisfies Property CS3. Analogously define $p(l_2)$. Then, segment $\overline{p(l_1)p(l_2)}$ splits P into two disjoint convex polygons P' and P'', where P' contains all and only the vertices in ν and P'' contains all and only the vertices not in ν, a s Γ satisfies Property CS3. By the convexity of P' and P'', e is internal to P'', while the part of ν inside P is internal to P'. Hence, e does not cross ν.

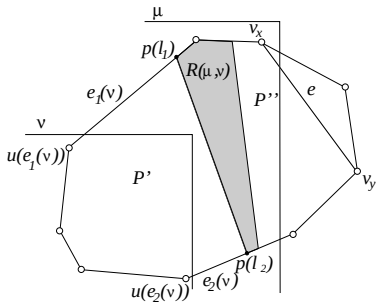


Figure 10.8: Drawing Γ_1 has no crossing between an edge e and a cluster containing none of the endvertices of e. The thick line represents segment $\overline{p(l_1)p(l_2)}$.

We prove that Γ_1 satisfies Property CS2. First, observe that the angles incident to v_i and v_j in P^1 are strictly less than 180°, since they are strictly less than 180° in Γ; further, the angles incident to v_x and v_y in P^1 are strictly less than 180°, since they are strictly less than the angles incident to v_x and v_y in P, that are at most 180°, by the convexity of P. Suppose that $\sigma(v_x)$ precedes $\sigma(v_y)$ in Σ, the opposite case being analogous. It suffices to observe that: (i) if C^1 contains both v_i and v_j, then $\Sigma'_1 = \Sigma$; hence, v_i and v_j are vertices satisfying the desired properties; (ii) if C^1 contains neither v_i nor v_j,

then Σ'_1 is the subsequence of Σ that starts at $\sigma(v_x)$ and ends at $\sigma(v_y)$; hence, v_x and v_y are vertices satisfying the desired properties; (iii) if C^1 contains v_j and does not contain v_i (as in Fig. 10.7), then Σ'_1 is the subsequence of Σ that starts at $\sigma(v_x)$ and ends at $\sigma(v_j) = \mu_k$; hence, v_x and v_j are vertices satisfying the desired properties.

We prove that Γ_1 satisfies Property CS3. The existence of regions $R(\mu, \nu)$ inside P^1, for every cluster ν, is easily deduced from the existence of regions $R(\mu, \nu)$ inside P, which is guaranteed as Γ satisfies Property CS3. Namely, if $R(\mu, \nu)$ is not intersected by e, then a region $R(\mu, \nu)$ inside P^1 can be constructed coincident with the same region inside P. Further, if $R(\mu, \nu)$ is cut by e, then two regions are created, one inside P^1 and the other one inside P^2. The properties that have to be satisfied by $R(\mu, \nu)$ inside P^1 easily descend from the analogous properties satisfied by $R(\mu, \nu)$ inside P. □

When dealing with outerclustered graphs and general clustered graphs, it is sufficient to consider clustered graphs whose underlying graphs have triangular outer faces. This leads us to the following refined definition (see Fig. 10.9):

Definition 10.4 *Let $C(G, T)$ be a clustered graph such that G is a 3-cycle (u, v, z). A straight-line rectangular drawing $\Gamma(C)$ of C is a triangular-convex-separated drawing if:*

- *TCS1: for every pair of clusters μ and ν such that μ is the parent of ν in T and such that μ is not an ancestor of $\sigma(u, v, z)$, there exists a convex region $R(\mu, \nu)$ such that: (i) $R(\mu, \nu)$ is entirely contained inside $\mu \cap (P \cup int(P))$, where P is the triangle representing G in $\Gamma(C)$; (ii) for any cluster $\mu' \neq \mu$ and any child ν' of μ', $R(\mu, \nu)$ intersects neither $R(\mu', \nu')$ nor the boundary of μ'; (iii) $R(\mu, \nu) \cap P$ consists of two polygonal lines $l_1(\mu, \nu)$ and $l_2(\mu, \nu)$ such that at least one endpoint of $l_1(\mu, \nu)$ (resp. of $l_2(\mu, \nu)$) belongs to $e_1(\nu)$ (resp. to $e_2(\nu)$).*

The interplay between a triangular-convex-separated drawing of a clustered 3-cycle and a convex-separated drawing of a linearly-ordered maximal outerclustered graph is clarified in the following lemma.

Lemma 10.4 *Let $C(G, T)$ be a linearly-ordered maximal outerclustered graph. Then, a triangular-convex-separated drawing of C_o is a convex-separated drawing of C_o.*

Proof: Consider any triangular-convex-separated drawing $\Gamma(C_o)$ of C_o. We prove that $\Gamma(C_o)$ is a convex-separated drawing. Properties CS1 and CS2 of

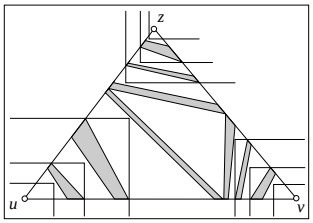


Figure 10.9: A triangular-convex-separated drawing of a clustered graph whose underlying graph is a 3-cycle.

Definition 10.3 easily descend from the fact that, since $o(G)$ is a 3-cycle, it is represented as a convex polygon P whose internal angles are smaller than 180°. Property CS3 coincides with Property TCS1. □

Finally, we define a class of drawings in which the properties of convexity and visibility among vertices and clusters are imposed on all the internal faces rather than on the outer face.

Definition 10.5 *Let $C(G,T)$ be an internally-triangulated clustered graph. A drawing $\Gamma(C)$ of C is an internally-convex-separated drawing if, for every internal face f of G, the part $\Gamma(C_f)$ of $\Gamma(C)$ representing C_f is a triangular-convex-separated drawing.*

10.3 How to Draw Linearly-Ordered Outerclustered Graphs

In this section we show how to construct an internally-convex-separated drawing of any linearly-ordered outerclustered graph C for an arbitrary triangular-convex-separated drawing of the outer face C_o of C. This is done by means of an inductive algorithm that uses the following lemma as the main tool (see Fig. 10.10):

Lemma 10.5 *Let $C(G,T)$ be an internally-triangulated triconnected outerclustered graph. Suppose that C is linearly-ordered according to a sequence $\mu_1, \mu_2, \ldots, \mu_k$ of clusters of T. Let v_i and v_j be any two vertices such that $\sigma(v_i) = \mu_1$ and $\sigma(v_j) = \mu_k$. Let V_1 (resp. V_2) be the set of vertices between v_i and v_j (resp. between v_j and v_i) in the clockwise order of the vertices around $o(G)$. Then, if $V_1 \neq \emptyset$, there exists a path $\mathcal{P}_u = (u_1, u_2, \ldots, u_r)$ such that:*

- *P1:* u_1 *and* u_r *belong to* $V_2 \cup \{v_i, v_j\}$*;*
- *P2:* u_i *is an internal vertex of* G*, for each* $2 \leq i \leq r-1$*;*
- *P3: if* $\sigma(u_i) = \mu_{j_1}$ *and* $\sigma(u_{i+1}) = \mu_{j_2}$*, then* $j_1 < j_2$*, for each* $1 \leq i \leq r-1$*;*
- *P4: there exists exactly one vertex* u_x*, where* $2 \leq x \leq r-1$*, that is adjacent to at least one vertex* v_x *in* V_1*;*
- *P5: there exist no chord among the vertices of path* $(u_1, u_2, \ldots, u_x)$ *and no chord among the vertices of path* $(u_x, u_{x+1}, \ldots, u_r)$*.*

Proof: We derive path $\mathcal{P}_u$ in several steps. At step $s+1$ a path $\mathcal{P}^{s+1}$ is found by modifying the path $\mathcal{P}^s$ obtained at the previous step. At the last step m of the algorithm, a path $\mathcal{P}_u$ can be obtained as a subpath of $\mathcal{P}^m$ satisfying the properties required by the lemma. A path satisfying properties P1–P5 is shown in Fig. 10.10.

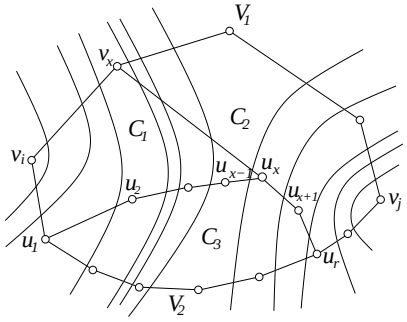


Figure 10.10: A path satisfying properties P1–P5.

More in detail, at each step $s \leq m$, a path $\mathcal{P}^s = (u_1^s, u_2^s, \ldots, u_{r(s)}^s)$ is found satisfying the properties described below (see Fig. 10.11).

Denote by $\mathcal{P}_1 = (v_i, v_{i+1}, \ldots, v_{j-1}, v_j)$ and $\mathcal{P}_2 = (v_i, v_{i-1}, \ldots, v_{j+1}, v_j)$ the monotone paths on the vertices of $V_1 \cup \{v_i, v_j\}$ and of $V_2 \cup \{v_i, v_j\}$, respectively, delimiting $o(G)$.

- PP1: $u_1^s = v_i$ and $u_{r(s)}^s = v_j$;
- PP2: each vertex u_t^s is either an internal vertex of G or a vertex of $\mathcal{P}_2$, for each $2 \leq t \leq r(s) - 1$;

- PP3: if $\sigma(u_t) = \mu_{j_1}$ and $\sigma(u_{t+1}) = \mu_{j_2}$, then $j_1 \leq j_2$, for each $1 \leq t \leq r(s) - 1$; and
- PP4: there exists no chord inside cycle $\mathcal{P}^s \cup \mathcal{P}_1$.

At the first step, set $\mathcal{P}^1 = \mathcal{P}_2$. Path $\mathcal{P}^1$ satisfies Property PP1 and PP2 by definition, Property PP3 since C satisfies Property LO2 of Definition 10.2, and Property PP4 because G is triconnected.

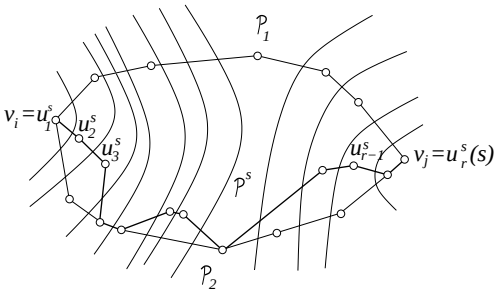


Figure 10.11: A path $\mathcal{P}^s$ satisfying properties PP1–PP4.

Suppose that a path $\mathcal{P}^s$ has been found at step s of the algorithm. We show how to determine $\mathcal{P}^{s+1}$ at step $s+1$ of the algorithm.

Consider any edge (u_t^s, u_{t+1}^s) of $\mathcal{P}^s$. Such an edge is incident to a face internal to cycle $\mathcal{P}^s \cup \mathcal{P}_1$. Let z_1^* be the third vertex of such a face. As $\mathcal{P}^s$ satisfies Property PP4, z_1^* is neither a vertex of $\mathcal{P}_1$ nor a vertex of $\mathcal{P}^s$. Hence, z_1^* is internal to $\mathcal{P}^s \cup \mathcal{P}_1$.

Denote by u_a^s and by u_b^s the first and the last vertex of $\mathcal{P}^s$ adjacent to z_1^*. We distinguish two cases.

In the first case, $\sigma(u_a^s)$, $\sigma(z_1^*)$, and $\sigma(u_b^s)$ appear in this order in Σ. Then, path $\mathcal{P}^{s+1}$ is obtained by replacing the subpath of $\mathcal{P}^s$ between u_a^s and u_b^s with path (u_a^s, z_1^*, u_b^s). An example of this case is shown in Fig. 10.12.

Suppose that z_1^* is adjacent to at least one vertex in V_1. Since the first and the last vertex of $\mathcal{P}^{s+1}$ (that are v_i and v_j, respectively) belong to $\mathcal{P}_2$, the vertices shared by $\mathcal{P}^{s+1}$ and $\mathcal{P}_2$ partition $\mathcal{P}^{s+1}$ into subpaths, where the internal vertices of each subpath are internal to $\mathcal{P}_1 \cup \mathcal{P}_2$. Let $\mathcal{P}_u$ be the one of such subpaths containing z_1^*. Path $\mathcal{P}_u = (u_1, u_2, \ldots, u_x, u_{x+1}, \ldots, u_r)$ defined as above is easily shown to satisfy properties P1–P4 (where $u_x = z_1^*$). In particular, notice that $\mathcal{P}_u$ satisfies Property P3 since $\mathcal{P}^{s+1}$ satisfies Property PP3

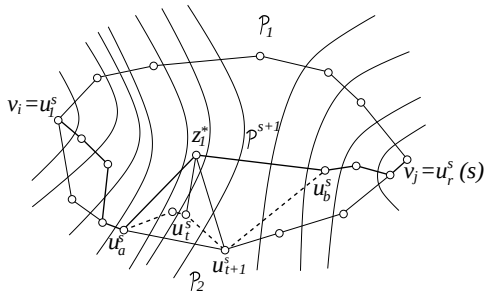


Figure 10.12: Obtaining $\mathcal{P}^{s+1}$ from $\mathcal{P}^s$ if $\sigma(u_a^s)$, $\sigma(z_1^*)$, and $\sigma(u_b^s)$ appear in this order in Σ. Dashed lines show the subpath of $\mathcal{P}^s$ not belonging to $\mathcal{P}^{s+1}$.

and no two adjacent vertices of $\mathcal{P}_u$ have the same smallest containing cluster, as C satisfies Property O3 of Definition 10.1. Further, if there exists a chord (u_a, u_b) among the vertices of path $(u_1, u_2, \ldots, u_x)$ (resp. among the vertices of path $(u_x, u_{x+1}, \ldots, u_r)$), then replace the subpath of $(u_1, u_2, \ldots, u_x)$ (resp. of $(u_x, u_{x+1}, \ldots, u_r)$) between u_a and u_b with such a chord. The repetition of such an argument eventually leads to a path $\mathcal{P}_u$ also satisfying Property P5. Hence, the algorithm stops because a path with the properties required by the lemma has been found.

Suppose that z_1^* is not adjacent to any vertex in V_1. Then, $\mathcal{P}^{s+1}$ is easily shown to satisfy properties PP1–PP4. Hence, the algorithm continues with step $s+2$.

In the second case, either $\sigma(z_1^*)$, $\sigma(u_a^s)$, and $\sigma(u_b^s)$ appear in this order in Σ, or they appear in the order $\sigma(u_a^s)$, $\sigma(u_b^s)$, $\sigma(z_1^*)$. Suppose that $\sigma(z_1^*)$, $\sigma(u_a^s)$, and $\sigma(u_b^s)$ appear in this order in Σ, the other case being analogous. See Fig. 10.13.

Consider edge (u_{a-1}^s, u_a^s) of $\mathcal{P}^s$. Such an edge is incident to a face internal to cycle $\mathcal{P}^s \cup \mathcal{P}_1$. Let z_2^* be the third vertex of such a face. By Property PP4, z_2^* is neither a vertex of $\mathcal{P}_1$ nor a vertex of $\mathcal{P}^s$. Hence, z_2^* is internal to $\mathcal{P}^s \cup \mathcal{P}_1$. Further, $\sigma(z_2^*)$ is not the same cluster of $\sigma(u_a^s)$ (since C satisfies Property O3 of Definition 10.1), and $\sigma(z_2^*)$ does not follow $\sigma(u_a^s)$ in Σ, otherwise edge (u_a^s, z_2^*) would cross twice the boundary of $\sigma(u_a^s)$ or the boundary of the cluster coming before $\sigma(u_a^s)$ in Σ (depending on whether $\sigma(u_a^s)$ is a child or is the parent of the cluster coming before $\sigma(u_a^s)$ in Σ). Hence, $\sigma(z_2^*)$ precedes $\sigma(u_a^s)$ in Σ.

Then, the whole argument can be repeated, namely denote by u_c^s and by u_d^s the first and the last vertex adjacent to z_2^* in $\mathcal{P}^s$. Since C satisfies Property O3, then $\sigma(u_c^s) \neq \sigma(z_2^*)$; since $\sigma(z_2^*)$ precedes $\sigma(u_a^s)$ in Σ and since $\sigma(u_a^s)$ does

not follow $\sigma(u_d^s)$ in Σ, then $\sigma(z_2^*)$ precedes $\sigma(u_d^s)$ in Σ. Then, either $\sigma(u_c^s)$ precedes $\sigma(z_2^*)$ in Σ or $\sigma(z_2^*)$ precedes $\sigma(u_c^s)$ in Σ. If $\sigma(u_c^s)$ precedes $\sigma(z_2^*)$, then a path $\mathcal{P}^{s+1}$ is found by replacing the subpath of $\mathcal{P}^s$ between u_c^s and u_d^s with path (u_c^s, z_2^*, u_d^s); then, either a path $\mathcal{P}_u$ satisfying properties P1–P5 can be derived from $\mathcal{P}^{s+1}$ or $\mathcal{P}^{s+1}$ satisfies properties PP1–PP4, depending on whether z_2^* is adjacent to at least one vertex in V_1 or not. If $\sigma(z_2^*)$ precedes $\sigma(u_c^s)$, then edge (u_{c-1}^s, u_c^s) and the vertex z_3^* incident to the face $(u_{c-1}^s, u_c^s, z_3^*)$ internal to cycle $\mathcal{P}^s \cup \mathcal{P}_1$ are considered and the argument is repeated again. The repetition of such an argument eventually leads to find a vertex z_f^* that is incident to a face $(u_{y-1}^s, u_y^s, z_f^*)$ and such that, denoting by u_p^s and u_q^s the first and the last neighbor of z_f^* in $\mathcal{P}^s$, $\sigma(u_p^s)$, $\sigma(z_f^*)$, and $\sigma(u_q^s)$ appear in this order in Σ. Namely, at every repetition of such an argument, the considered edge (that is equal to (u_t^s, u_{t+1}^s) at the first repetition, to (u_{a-1}^s, u_a^s) at the second repetition, to (u_{c-1}^s, u_c^s) at the third repetition, and to (u_{y-1}^s, u_y^s) at the last repetition) gets closer to v_i in $\mathcal{P}^s$; then, after a certain number of repetitions of the algorithm, vertex u_{y-1}^s eventually belongs to cluster μ_1 and no cluster precedi ng μ_1 exists in Σ.

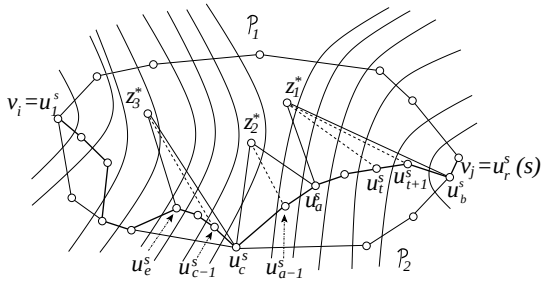


Figure 10.13: Three repetitions of the argument for the case in which $\sigma(z_1^*)$ (resp. $\sigma(z_2^*)$ and $\sigma(z_3^*)$) precedes the smallest clusters containing the first and last neighbor of z_1^* (resp. of z_2^* or of z_3^*) in $\mathcal{P}^s$. Thick segments represent $\mathcal{P}^s$.

It remains to observe that, after a certain number m of steps of the algorithm, a path $\mathcal{P}_u$ satisfying properties P1–P5 can be derived from $\mathcal{P}^m$. Namely, the number of vertices internal to cycle $\mathcal{P}^s \cup \mathcal{P}_1$ decreases at every step of the algorithm, hence a vertex z^* adjacent to a vertex in V_1 is eventually added to a path $\mathcal{P}^{m-1}$ to form a path $\mathcal{P}^m$, from which a path $\mathcal{P}_u$ satisfying properties P1–P5 can then be derived. Notice that z^* is the only vertex of $\mathcal{P}^m$ adjacent to a vertex in V_1, since no vertex of $\mathcal{P}^{m-1}$ is adjacent to a vertex in V_1, as

$\mathcal{P}^{m-1}$ satisfies Property PP4. □

A pseudo-code description of the algorithm for finding a path $\mathcal{P}_u$ satisfying the properties required by Lemma 10.5 (supposing that $V_1 \neq \emptyset$) is presented in Algorithm 7.

A lemma similar to Lemma 10.5 is presented in the following. The proof of such a lemma can be obtained analogously to the one of Lemma 10.5, where V_1 replaces V_2 and vice versa.

Lemma 10.6 *Let $C(G,T)$ be an internally-triangulated triconnected outerclustered graph. Suppose that C is linearly-ordered according to a sequence $\mu_1, \mu_2, \ldots, \mu_k$ of clusters of T. Let v_i and v_j be any two vertices such that $\sigma(v_i) = \mu_1$ and $\sigma(v_j) = \mu_k$. Let V_1 (resp. V_2) be the set of vertices between v_i and v_j (resp. between v_j and v_i) in the clockwise order of the vertices around $o(G)$. Then, if $V_2 \neq \emptyset$, there exists a path $\mathcal{P}_u = (u_1, u_2, \ldots, u_r)$ such that:*

- *P1: u_1 and u_r belong to $V_1 \cup \{v_i, v_j\}$;*
- *P2: u_i is an internal vertex of G, for each $2 \leq i \leq r-1$;*
- *P3: if $\sigma(u_i) = \mu_{j_1}$ and $\sigma(u_{i+1}) = \mu_{j_2}$, then $j_1 < j_2$, for each $1 \leq i \leq r-1$;*
- *P4: there exists exactly one vertex u_x, where $2 \leq x \leq r-1$, that is adjacent to at least one vertex v_x in V_2;*
- *P5: there exist no chord among the vertices of path $(u_1, u_2, \ldots, u_x)$ and no chord among the vertices of path $(u_x, u_{x+1}, \ldots, u_r)$.*

We now present the main theorem of this section.

Theorem 10.1 *Let $C(G,T)$ be a linearly-ordered internally-triangulated triconnected outerclustered graph. Then, for every convex-separated drawing $\Gamma(C_o)$ of C_o, there exists an internally-convex-separated drawing $\Gamma(C)$ of C completing $\Gamma(C_o)$.*

Proof: We prove the statement by induction on the number of internal vertices of G. First observe that, since G is triconnected, $o(G)$ has no chords. Hence, if G has no internal vertices, then C_o and C are the same graph and the statement trivially follows. Otherwise, G has internal vertices. Denote by $\mathcal{C}$ the cycle delimiting $o(G)$. We show how to split C into smaller linearly-ordered internally-triangulated triconnected outerclustered graphs.

Algorithm 7 PATHS IN LINEARLY-ORDERED OUTERCLUSTERED GRAPHS

Require: $C(G,T)$, Σ, $V_1 \neq \emptyset$, $\mathcal{P}_1$, and $\mathcal{P}_2$, as in Lemma 10.5.
Ensure: A path $\mathcal{P}_u = (u_1, u_2, \ldots, u_r)$ satisfying Properties P1–P5 of Lemma 10.5.

1: $s \leftarrow 1$; $\mathcal{P}^s \leftarrow \mathcal{P}_2$; *path_found* $\leftarrow$ FALSE; $e \leftarrow$ any edge (u_t^s, u_{t+1}^s) of $\mathcal{P}^s$;
2: **while** *path_found* = FALSE **do**
3: $i \leftarrow 1$;
4: $z_i^* \leftarrow$ the vertex creating a face with e inside $\mathcal{P}^s \cup \mathcal{P}_1$;
5: $u_L^s \leftarrow$ the first vertex of $\mathcal{P}^s$ adjacent to z_i^*;
6: $u_R^s \leftarrow$ the last vertex of $\mathcal{P}^s$ adjacent to z_i^*;
7: **if** $\sigma(u_L^s)$, $\sigma(z_i^*)$, and $\sigma(u_R^s)$ do not appear in this order in Σ **then**
8: **while** $\sigma(z_i^*)$, $\sigma(u_L^s)$, and $\sigma(u_R^s)$ appear in this order in Σ **do**
9: $e \leftarrow$ edge (u_{L-1}^s, u_L^s) of $\mathcal{P}^s$; $i \leftarrow i+1$;
10: $z_i^* \leftarrow$ the vertex creating a face with e inside $\mathcal{P}^s \cup \mathcal{P}_1$;
11: $u_L^s \leftarrow$ the first vertex of $\mathcal{P}^s$ adjacent to z_i^*;
12: $u_R^s \leftarrow$ the last vertex of $\mathcal{P}^s$ adjacent to z_i^*;
13: **end while**
14: **while** $\sigma(u_L^s)$, $\sigma(u_R^s)$, and $\sigma(z_i^*)$ appear in this order in Σ **do**
15: $e \leftarrow$ edge (u_R^s, u_{R+1}^s) of $\mathcal{P}^s$; $i \leftarrow i+1$;
16: $z_i^* \leftarrow$ the vertex creating a face with e inside $\mathcal{P}^s \cup \mathcal{P}_1$;
17: $u_L^s \leftarrow$ the first vertex of $\mathcal{P}^s$ adjacent to z_i^*;
18: $u_R^s \leftarrow$ the last vertex of $\mathcal{P}^s$ adjacent to z_i^*;
19: **end while**
20: **end if**
21: $z^* \leftarrow z_i^*$;
22: $\mathcal{P}^{s+1} \leftarrow$ the path obtained from $\mathcal{P}^s$ by replacing the subpath between u_L^s and u_R^s with path (u_L^s, z^*, u_R^s);
23: **if** z^* is adjacent to at least one vertex in V_1 **then**
24: $m \leftarrow s+1$;
25: partition $\mathcal{P}^m$ into subpaths, based on the vertices of $\mathcal{P}^m \cap \mathcal{P}_2$;
26: $\mathcal{P}_u \leftarrow$ the subpath of $\mathcal{P}^m$ containing z^*;
27: **while** $(u_1, u_2, \ldots, u_x = z^*)$ has a chord (u_a, u_b) **do**
28: replace the subpath of $\mathcal{P}_u$ between u_a and u_b with (u_a, u_b);
29: **end while**
30: **while** $(u_x = z^*, u_{x+1}, \ldots, u_r)$ has a chord (u_a, u_b) **do**
31: replace the subpath of $\mathcal{P}_u$ between u_a and u_b with (u_a, u_b);
32: **end while**
33: *path_found* $\leftarrow$ TRUE;
34: **else**
35: $s \leftarrow s+1$; $e \leftarrow$ any edge (u_t^s, u_{t+1}^s) of $\mathcal{P}^s$;
36: **end if**
37: **end while**
38: **return** $\mathcal{P}_u$;

Let $\Gamma(C_o)$ be an arbitrary convex-separated drawing of C_o. Denote by P the polygon representing $o(G)$ in $\Gamma(C_o)$. Since $\Gamma(C_o)$ satisfies Property CS1 of Definition 10.3, P is convex. Suppose that C is linearly-ordered according to a sequence $\Sigma = \mu_1, \mu_2, \ldots, \mu_h, \ldots, \mu_k$ of clusters of T. Let v_i and v_j be two vertices of $\mathcal{C}$ with $\sigma(v_i) = \mu_1$ and $\sigma(v_j) = \mu_k$ such that the angle of P incident to v_i and the angle of P incident to v_j are strictly less than 180°. Such vertices exist since $\Gamma(C_o)$ satisfies Property CS2 of Definition 10.3.

We distinguish two cases:

Case 1 applies when the vertices of $V_1 \cup \{v_i, v_j\}$ are not all collinear. In such a case, $V_1 \neq \emptyset$, otherwise the only two vertices of $V_1 \cup \{v_i, v_j\}$ would be collinear. Hence, a path $(u_1, u_2, \ldots, u_x, \ldots, u_r)$ can be found as in Lemma 10.5. Let v_x be any vertex of V_1 adjacent to u_x.

Lemma 10.7 *Vertex v_x does not lie on the line $l(u_1, u_r)$ through u_1 and u_r.*

Proof: Denote by $\mathcal{P}(u_1, v_x, u_r)$ the subpath of $\mathcal{C}$ between vertices u_1 and u_r containing v_x. Such a path is hence part of the boundary of $o(G)$. If there exists an internal vertex of $\mathcal{P}(u_1, v_x, u_r)$ lying on the segment $\overline{u_1 u_r}$, then all the vertices in $\mathcal{P}(u_1, v_x, u_r)$ lie on $\overline{u_1 u_r}$, otherwise polygon P would not be convex (see Fig. 10.14(a)). However, $\mathcal{P}(u_1, v_x, u_r)$ contains all the vertices of $V_1 \cup \{v_i, v_j\}$, that are not all collinear, by hypothesis. Now suppose that v_x lies on the half-line that is part of $l(u_1, u_r)$, that starts at u_1, and that does not contain u_r (the case in which v_x lies on the half-line that is part of $l(u_1, u_r)$, that starts at u_r, and that does not contain u_1 being analogous). By the convexity of P, all the vertices in the path between v_x and u_r containing u_1 lie on $l(u_1, u_r)$ (see Fig. 10.14(b)). However, since $v_x \in V_1$ and since v_i and v_j are the first and the last vertex in the path between u_1 and u_r not containing v_x, it follows that v_i is one of the internal vertices of such a path, thus its incident angle in P is exactly 180°, contradicting the hypothesis. □

Consider the triangle $T(v_x, u_1, u_r)$ with vertices v_x, u_1, and u_r. By Lemma 10.7, such a triangle is non-degenerate and, by the convexity of P, it is entirely contained inside P (observe that parts of triangle $T(v_x, u_1, u_r)$ and of polygon P could coincide). Denote by $\sigma'(u_i)$ any cluster in Σ which is a child of $\sigma(u_i)$, for each $2 \leq i \leq r-1$.

Lemma 10.8 *There exists a small disk D entirely contained inside $int(T(v_x, u_1, u_r)) \cap R(\sigma(u_x), \sigma'(u_x))$.*

Proof: Let μ_{j_1}, μ_{j_2}, μ_{j_3}, and μ_{j_4} be clusters $\sigma(u_1)$, $\sigma(u_x)$, $\sigma(u_r)$, and $\sigma(v_x)$ in Σ, respectively. Denote by μ'_j any cluster in Σ which is a child of μ_j. Since

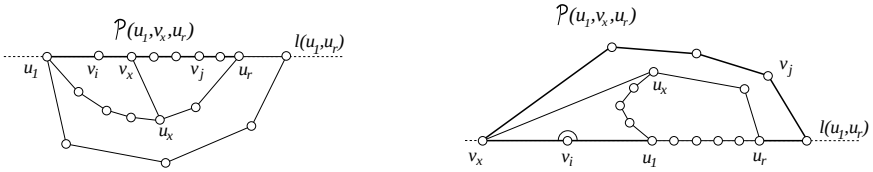


Figure 10.14: Illustration for the proof of Lemma 10.7. (a) Vertex v_x lies on $\overline{u_1u_r}$. (b) Vertex v_x lies on the half-line that is part of $l(u_1, u_r)$, that starts at u_1, and that does not contain u_r.

$2 \leq x \leq r-1$ and since $\sigma(u_i) = \mu_{l_1}$ and $\sigma(u_{i+1}) = \mu_{l_2}$ implies $l_1 < l_2$, we have $j_1 < j_2 < j_3$. Since C satisfies Property O3 of Definition 10.1, $\sigma(u_x) \neq \sigma(v_x)$ and hence $j_2 \neq j_4$. Suppose that $j_2 < j_4$, the case $j_2 > j_4$ being analogous.

We claim that $s_1 = R(\mu_{j_2}, \mu'_{j_2}) \cap \overline{u_1u_r}$ and $s_2 = R(\mu_{j_2}, \mu'_{j_2}) \cap \overline{u_1v_x}$ are straight-line segments. The claim implies the lemma, as if the claim holds, then the interior of the quadrilateral having s_1 and s_2 as opposite sides entirely belongs to $int(T(v_x, u_1, u_r)) \cap R(\mu_{j_2}, \mu'_{j_2})$ and any disk entirely contained inside such a quadrilateral satisfies the requirements of the lemma. See Fig. 10.15.

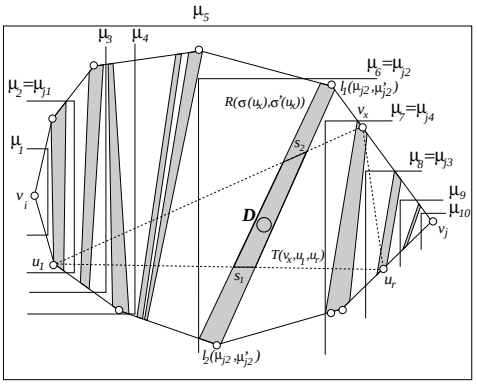


Figure 10.15: Illustration for the proof of Lemma 10.8.

By Property CS3 of Definition 10.3, $R(\mu_{j_2}, \mu'_{j_2}) \cap P$ consists of two line segments $l_1(\mu_{j_2}, \mu'_{j_2})$ and $l_2(\mu_{j_2}, \mu'_{j_2})$. Since $j_2 \neq j_1, j_3, j_4$, none of vertices u_1, u_r, and v_x belongs to $R(\mu_{j_2}, \mu'_{j_2})$. Consider the two line segments l_a and l_b

obtained by removing $l_1(\mu_{j_2}, \mu'_{j_2})$ and $l_2(\mu_{j_2}, \mu'_{j_2})$ from P. Since P is convex, it suffices to show that u_1 is in l_a and u_r is in l_b, or vice versa, in order to prove that s_1 is a straight-line segment; analogously, it suffices to show that u_1 is in l_a and v_x is in l_b, or vice versa, in order to prove that s_2 is a straight-line segment. However, this follows from the fact that one of the sides of $R(\mu_{j_2}, \mu'_{j_2})$ separates the vertices whose smallest containing cluster is μ_j, with $j < j_2$, from the vertices whose smallest containing cluster is μ_j, with $j > j_2$. □

Place u_x at any point inside D. Draw edge (u_x, v_x) as a straight-line segment. Consider the straight-line segment $\overline{u_1 u_x}$ and consider any cluster μ_j such that $j_1 < j < j_2$. Then, $R(\mu_j, \mu'_j) \cap \overline{u_1 u_x}$ is a straight-line segment, namely one of the sides of region $R(\mu_j, \mu'_j)$ separates the vertices whose smallest containing cluster is μ_{j^*}, with $j^* < j$, from the vertices whose smallest containing cluster is μ_{j^*}, with $j^* > j$. Analogously, consider the straight-line segment $\overline{u_x u_r}$ and consider any cluster μ_j such that $j_2 < j < j_3$. Then, $R(\mu_j, \mu'_j) \cap \overline{u_x u_r}$ is a straight-line segment, namely one of the sides of region $R(\mu_j, \mu'_j)$ separates the vertices whose smallest containing cluster is μ_{j^*}, with $j^* < j$, from the vertices whose smallest containing cluster is μ_{j^*}, with $j^* > j$. Draw each vertex u_i, with $2 \leq i \leq x-1$, at any internal point of $\overline{u_1 u_x} \cap R(\sigma(u_i), \sigma'(u_i))$. Analogously, draw each vertex u_i, with $x+1 \leq i \leq r-1$, at any point of $\overline{u_x u_r} \cap R(\sigma(u_i), \sigma'(u_i))$. Denote by Γ the constructed drawing. The construction of Γ is depicted in Fig. 10.16.

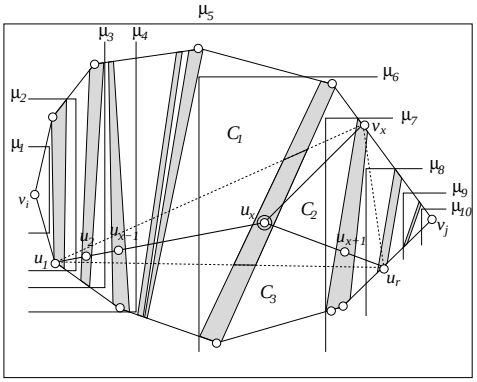


Figure 10.16: Drawing path $(u_1, u_2, \ldots, u_x, \ldots, u_r)$ and edge (u_x, v_x).

Let $\mathcal{C}_1$ be the cycle composed of path $(u_1, u_2, \ldots, u_x)$, of edge (u_x, v_x), and of the path between u_1 and v_x in $\mathcal{C}$ not containing u_r. Let $\mathcal{C}_2$ be the

cycle composed of path $(u_x, u_{x+1}, \ldots, u_r)$, of edge (u_x, v_x), and of the path between v_x and u_r in $\mathcal{C}$ not containing u_1. Let $\mathcal{C}_3$ be the cycle composed of path $(u_1, u_2, \ldots, u_x, \ldots, u_r)$ and of the path between u_1 and u_r in $\mathcal{C}$ not containing v_x. Let $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ be the subtrees of T induced by the clusters containing vertices of $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$, respectively. Denote by G_1, G_2, and G_3 the subgraphs of G induced by the vertices inside or on the border of $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$, respectively. Finally, let C_1, C_2, and C_3 be the clustered graphs whose underlying graphs are G_1, G_2, and G_3, respectively, and whose inclusion trees T_1, T_2, and T_3 are the subtrees of T induced by the clusters containing vertices of G_1, G_2, and G_3, respectively.

Lemma 10.9 *C_1, C_2, and C_3 are linearly-ordered outerclustered graphs.*

Proof: Denote by $\mathcal{C}_{1,2}$ the cycle composed of path $(u_1, u_2, \ldots, u_x, \ldots, u_r)$ and of the path between u_1 and u_r in $\mathcal{C}$ containing v_x. Let $C_{1,2}$ be the clustered graph whose underlying graph $G_{1,2}$ is the subgraph of G whose vertices are incident to or internal to $\mathcal{C}_{1,2}$, and whose inclusion tree $T_{1,2}$ is the subtree of T induced by the clusters containing vertices of $G_{1,2}$. It suffices to prove that $C_{1,2}$ and C_3 are linearly-ordered outerclustered graphs. Namely, by Lemma 10.2, if $C_{1,2}$ is a linearly-ordered outerclustered graph, then C_1 and C_2 are linearly-ordered outerclustered graphs, as well.

Since C is linearly-ordered, $o(G)$ is delimited by two monotone paths $\mathcal{P}_1$ and $\mathcal{P}_2$, where $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) has the vertices in V_1 (resp. in V_2) as internal vertices.

Consider the sequence $\Sigma = \mu_1, \mu_2, \ldots, \mu_k$ according to which C is linearly-ordered. Denote by Σ_3 the subsequence of Σ that starts at $\sigma(u_1)$ and ends at $\sigma(u_r)$. We prove that $C_{1,2}$ is an outerclustered graph linearly-ordered according to Σ and that C_3 is an outerclustered graph linearly-ordered according to Σ_3.

Each cluster μ_i that is not crossed by path $(u_1, u_2, \ldots, u_x, \ldots, u_r)$ has two intersections with $\mathcal{C}_{1,2}$ and none with $\mathcal{C}_3$. Consider any cluster μ_i that is crossed by path $(u_1, u_2, \ldots, u_x, \ldots, u_r)$. Since C satisfies Property O2, the boundary $B(\mu_i)$ of each cluster μ_i intersects $\mathcal{C}$ exactly twice. One of such intersections belongs to $\mathcal{C}_{1,2}$, the other one to $\mathcal{C}_3$. Hence, each of $\mathcal{C}_{1,2}$ and $\mathcal{C}_3$ crosses $B(\mu_i)$ exactly twice. Then, Lemma 10.1 ensures that $C_{1,2}$ and C_3 are biconnected internally-triangulated outerclustered graphs.

We prove that $C_{1,2}$ and C_3 satisfy Property LO2 of Definition 10.2. Path $\mathcal{P}_1$ and the path obtained by replacing the subpath of $\mathcal{P}_2$ between u_1 and u_r with path $(u_1, u_2, \ldots, u_x, \ldots, u_r)$ are monotone paths delimiting $o(G_{1,2})$.

Further, path $(u_1, u_2, \ldots, u_x, \ldots, u_r)$ and the subpath of $\mathcal{P}_2$ between u_1 and u_r are monotone paths delimiting $o(G_3)$.

We prove that $C_{1,2}$ and C_3 satisfy Property LO1. Since C is linearly-ordered, the smallest cluster containing each vertex in G (and each vertex in $G_{1,2}$) is a cluster μ_i, that by hypothesis belongs to Σ. Concerning C_3, observe that $o(G_3)$ is delimited by two monotone paths. Hence, if the smallest cluster containing a vertex in G_3 does not belong to Σ_3, then such a vertex is outside $o(G_3)$, a contradiction.

We prove that $C_{1,2}$ and C_3 satisfy Property LO3. Sequence Σ for $C_{1,2}$ coincides with sequence Σ for C, hence the property follows. Concerning C_3, the existence of an index h such that (i) μ_{i+1} is the parent of μ_i, for each $i < h$ such that μ_i belongs to Σ_3, and (ii) μ_{i+1} is a child of μ_i, for each $i \geq h$ such that μ_{i+1} belongs to Σ_3, easily descends from the existence of index h for the sequence Σ of C and from the fact that Σ_3 is a subsequence of Σ. □

We turn our attention to the geometry supporting the above topological results.

Lemma 10.10 *The constructed drawings of* $(\mathcal{C}_1, \mathcal{T}_1)$, $(\mathcal{C}_2, \mathcal{T}_2)$, *and* $(\mathcal{C}_3, \mathcal{T}_3)$ *are convex-separated drawings.*

Proof: Drawing Γ is straight-line and rectangular by construction. By construction, u_x lies in $int(T(v_x, u_1, u_r))$ that in turn is entirely contained in $int(P)$. By the convexity of P, straight-line segments can be drawn from u_x to any vertex of P (and hence to u_1, v_x, and u_r) not causing crossings; hence, the drawings of $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ have no edge crossings. Since $\Gamma(C_o)$ has no region-region crossings, Γ has no region-region crossings (namely, each cluster has the same drawing in Γ and in $\Gamma(C_o)$).

We prove that Γ has no edge-region crossing. Suppose that there is an edge-region crossing between an edge e and a cluster ν. Then, e is either an edge of path $(u_1, u_2, \ldots, u_x)$, or an edge of path $(u_x, u_{x+1}, \ldots, u_r)$, or edge (u_x, v_x). Namely, all other edge-cluster pairs have the same drawings in Γ and in $\Gamma(C_o)$, hence they do not cross more than once by hypothesis. Suppose that $e = (u_i, u_{i+1})$ is an edge of $(u_1, u_2, \ldots, u_x)$, the other cases being analogous. If both u_i and u_{i+1} belong to ν then, by the convexity of ν, e is internal to ν. If exactly one of u_i and u_{i+1} belongs to ν then, by the convexity of ν, e crosses ν exactly once. It follows that ν contains neither u_i nor u_{i+1}. Consider the parent μ of ν in T. Such a parent exists since otherwise ν would be the root of T, contradicting the fact that ν contains neither u_i nor u_{i+1}. By definition of convex-separated drawing, there exists a convex region $R(\mu, \nu)$ with the properties described in Definition 10.3 which separates ν from the rest of the

drawing, thus avoiding an edge-region crossing between e and ν. More precisely, by definition of outerclustered graph, ν has exactly two incident edges $e_1(\nu)$ and $e_2(\nu)$ belonging to $o(G)$. Denote by $u(e_1(\nu))$ and $u(e_2(\nu))$ the endvertices of $e_1(\nu)$ and $e_2(\nu)$ belonging to ν. Denote by $p(l_1)$ the endpoint of $l_1(\mu, \nu)$ that lies on $e_1(\nu)$ (if both endpoints of $l_1(\mu, \nu)$ lie on $e_1(\nu)$, then $p(l_1)$ is the one that is closer to $u(e_1(\nu))$). Note that an endpoint of $l_1(\mu, \nu)$ lying on $e_1(\nu)$ exists as Γ satisfies Property CS3. Analogously define $p(l_2)$. Then, segment $\overline{p(l_1)p(l_2)}$ splits P into two disjoint convex polygons P' and P'', where P' contains all and only the vertices of $o(G)$ and of path $(u_1, u_2, \ldots, u_x)$ in ν and P'' contains all and only the vertices of $o(G)$ and of path $(u_1, u_2, \ldots, u_x)$ not in ν, as Γ satisfies Property CS3. By the convexity of P' and P'', e is internal to P'', while the part of ν inside P is internal to P'. Hence, e does not cross ν.

We prove that Γ satisfies Property CS1 of Definition 10.3. Denote by P_i the polygon representing $\mathcal{C}_i$ in Γ, for each $i \in \{1, 2, 3\}$. Every angle that is incident to a vertex in $\mathcal{C}$ different from u_1, u_r, and v_x in P_1, P_2, or P_3 is no more than 180°, since it is the same angle as in P. Every angle that is incident to a vertex in the path $(u_1, u_2, \ldots, u_r)$ different from u_1, u_x, and u_r in P_1, P_2, or P_3 is exactly 180°, by construction. Every angle incident to u_1, v_x, and u_r in P_1, P_2, or P_3 is strictly less than 180°, since it is strictly less than an angle that is at most 180° (namely the angle incident to the same vertex in P); finally, the three angles $\widehat{u_1u_xv_x}$, $\widehat{u_1u_xu_r}$, and $\widehat{u_ru_xv_x}$ incident to u_x are all strictly less than 180°, since they are angles in the non-degenerate triangles $T(u_1u_xv_x)$, $T(u_1u_xu_r)$, and $T(u_ru_xv_x)$, respectively.

We prove that Γ satisfies Property CS2. Concerning the drawing of $\mathcal{C}_1$ (the arguments for the drawing of $\mathcal{C}_2$ being analogous), observe that C_1 is linearly-ordered according to the subsequence Σ_1 of Σ that starts at $\sigma(v_i) = \mu_1$ and ends at the one of $\sigma(u_x)$ and $\sigma(v_x)$ that comes after in Σ. The angle incident to v_i in P_1 is strictly less than 180°, by hypothesis; further, as proved above, the angles incident to u_x and v_x in P_1 are strictly less than 180°. Concerning the drawing of $\mathcal{C}_3$, observe that $\sigma(u_1)$ and $\sigma(u_r)$ are the first and the last cluster in Σ_3. Further, the angles incident to u_1 and to u_r in P_3 are strictly smaller than 180°, as proved above.

We prove that Γ satisfies Property CS3. For each $i = 1, 2, 3$, let μ_j be any cluster (except for the first and the last one) belonging to the sequence according to which C_i is linearly-ordered and let μ'_j be any child of μ_j. The existence of a region $R(\mu_j, \mu'_j)$ inside P_i is easily deduced from the existence of region $R(\mu_j, \mu'_j)$ inside P. Namely, either $R(\mu_j, \mu'_j)$ inside P is intersected neither by path $(u_1, u_2, \ldots, u_r)$ nor by edge (u_x, v_x), implying that a region $R(\mu_j, \mu'_j)$ inside P_i can be constructed coincident with the same region inside

P, or $R(\mu_j, \mu'_j)$ inside P is intersected by one or both of path $(u_1, u_2, \ldots, u_r)$ and edge (u_x, v_x), that thus split $R(\mu_j, \mu'_j)$ into two or three regions inside the polygons P_i. The properties that have to be satisfied by $R(\mu_j, \mu'_j)$ inside P_i easily descend from the analogous properties satisfied by $R(\mu_j, \mu'_j)$ inside P. □

Graphs C_1, C_2, and C_3 are, in general, not triconnected; namely, there could exist chords between any vertex in $(u_1, u_2, \ldots, u_{x-1})$ and any vertex in $(u_{x+1}, u_{x+2}, \ldots, u_r)$, or chords between any vertex in $(u_1, u_2, \ldots, u_r)$ and any vertex in the path of $\mathcal{C}$ connecting u_1 and u_r and not containing v_x, or chords between u_x and any vertex in the path of $\mathcal{C}$ connecting u_1 and u_r and containing v_x. By Lemma 10.2, each of these chords splits a linearly-ordered outerclustered graph into two smaller linearly-ordered outerclustered graphs. Further, by construction the endvertices of each of such chords are not collinear with any other vertex of the cycle. Hence, by Lemma 10.3, drawing the chords as straight-line segments splits the drawings of C_1, C_2, and C_3, that are convex-separated by Lemma 10.10, into convex-separated drawings. When all the chords have been added, the underlying graphs of the resulting internally-triangulated linearly-ordered outerclustered graphs are all triconnected. Hence, the induction applies and an internally-convex-separated drawing of each of such linearly-ordered clustered graphs can be constructed inside the corresponding outer face, thus obtaining an internally-convex-separated drawing of C.

A pseudo-code description of the algorithm for drawing a linearly-ordered outerclustered graph C (supposing that $V_1 \neq \emptyset$) is presented in Algorithm 8.

Case 2 applies when the vertices of $V_2 \cup \{v_i, v_j\}$ are not all collinear. In such a case, $V_2 \neq \emptyset$ and a path $(u_1, u_2, \ldots, u_x, \ldots, u_r)$ can be found as in Lemma 10.6. Analogously to Case 1, C is decomposed into smaller triconnected internally-triangulated linearly-ordered outerclustered graphs; path $(u_1, u_2, \ldots, u_x, \ldots, u_r)$ and all the edges connecting vertices of such a path with vertices of $\mathcal{C}$ can be drawn so that the outer faces of the corresponding outerclustered graphs are represented by convex-separated drawings. Hence, the induction applies and an internally-convex-separated drawing of each of such linearly-ordered clustered graphs can be constructed inside the corresponding outer face, thus obtaining an internally-convex-separated drawing of C.

It remains to prove that one out of Case 1 and Case 2 applies. If Case 1 does not apply, then the vertices of $V_1 \cup \{v_i, v_j\}$ are collinear, and if Case 2 does not apply, then the vertices of $V_2 \cup \{v_i, v_j\}$ are collinear. However, this implies that all the vertices of $\mathcal{C}$ are collinear, contradicting the fact that $\Gamma(C_o)$ is a convex-separated drawing. This concludes the proof of Theorem 10.1. □

Algorithm 8 DRAWING LINEARLY-ORDERED OUTERCLUSTERED GRAPHS

Require: $C(G,T)$, $\Gamma(C_o)$, Σ, $V_1 \neq \emptyset$, $\mathcal{P}_1$, and $\mathcal{P}_2$.
Ensure: An internally-convex-separated drawing $\Gamma(C)$ of C completing $\Gamma(C_o)$.

1: $\mathcal{P}_u = (u_1, \ldots, u_x, \ldots, u_r) \leftarrow$ the path obtained from Algorithm 7 with input $C(G,T)$, Σ, $V_1 \neq \emptyset$, $\mathcal{P}_1$, and $\mathcal{P}_2$;
2: $\sigma'(u_i) \leftarrow$ any cluster in Σ child of $\sigma(u_i)$;
3: $v_x \leftarrow$ any vertex in V_1 adjacent to u_x;
4: $D \leftarrow$ disk in the interior of $T(v_x, u_1, u_r)$; place u_x at any point inside D;
5: **for** $2 \leq i \leq x-1$ **do**
6: place u_i at any internal point of $\overline{u_1 u_x} \cap R(\sigma(u_i), \sigma'(u_i))$;
7: **end for**
8: **for** $x+1 \leq i \leq r-1$ **do**
9: place u_i at any internal point of $\overline{u_x u_r} \cap R(\sigma(u_i), \sigma'(u_i))$;
10: **end for**
11: draw each edge between two vertices among the ones in $\mathcal{C}$ and in $\mathcal{P}_u$ as a straight-line segment;
12: apply Algorithm 8 on each resulting linearly-ordered outerclustered graph completing a drawing of the corresponding outer face;
13: $\Gamma(C) \leftarrow$ the resulting internally-convex-separated drawing of C;
14: **return** $\Gamma(C)$;

10.4 How to Draw Outerclustered Graphs

In this section we generalize from linearly-ordered outerclustered graphs to general outerclustered graphs. In order to show that any maximal outerclustered graph has an internally-convex-separated drawing completing an arbitrary triangular-convex-separated drawing of its outer face, we show how to reduce the problem of drawing an outerclustered graph to the one of drawing some linearly-ordered outerclustered graphs.

First, we need the following preliminary results.

Lemma 10.11 *Let $C(G,T)$ be a maximal outerclustered graph and let u, v, and z be the vertices incident to $o(G)$. Let $u_i \neq u$ be any internal vertex of G such that $\sigma(u_i)$ contains u and contains neither v nor z. Then, there exists an edge (u_i, u_{i+1}) in G such that $\sigma(u_{i+1})$ is a descendant of $\sigma(u_i)$.*

Proof: Refer to a c-planar embedding of C and to Fig. 10.17. Let μ_{i+1}

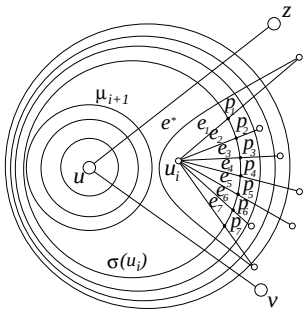


Figure 10.17: Illustration for the proof of Lemma 10.11.

be the only cluster child of $\sigma(u_i)$ and containing u. Notice that such a child exists. Namely, if such a child does not exist, then $\sigma(u_i) = \sigma(u)$; however, this would imply that $u_i = u$, since $\sigma(u)$ induces a connected graph, by the c-planarity and the maximality of C, and since an outerclustered graph does not contain two adjacent vertices having the same smallest containing cluster if one of them is internal, by Property O3 of Definition 10.1. Suppose, for a contradiction, that u_i is not adjacent to any vertex in μ_{i+1}. By Property O3, u_i is not adjacent to any vertex v_j such that $\sigma(v_j) = \sigma(u_i)$. It follows that each edge e_l incident to u_i intersects the boundary $B(\sigma(u_i))$ of $\sigma(u_i)$ in a point p_l belonging to the line segment $l(\sigma(u_i))$ that is the part of $B(\sigma(u_i))$ lying in the interior of $o(G)$. Order the points p_l as they are encountered walking on $l(\sigma(u_i))$ from one endpoint to the other one. Denote by $p_1, p_2, \ldots, p_m$ such points. By the maximality of G, $m \geq 3$ and there exists an internal face delimited by a 3-cycle containing edges e_1 and e_m. However, the third edge e^* of such a cycle crosses twice $l(\sigma(u_i))$ and hence $B(\sigma(u_i))$, contradicting the c-planarity of the considered embedding. □

Corollary 10.1 *Let $C(G,T)$ be any maximal outerclustered graph and let u, v, and z be the vertices incident to $o(G)$. Suppose that $\sigma(u) \neq \sigma(v), \sigma(z)$. Let u_1 be any internal vertex of G such that $\sigma(u_1)$ contains u and contains neither v nor z. Then, there exists a chordless path $(u_1, u_2, \ldots, u_k)$ in G such that $u_k = u$ and such that $\sigma(u_{i+1})$ is a descendant of $\sigma(u_i)$, for each $i = 1, 2, \ldots, k-1$.*

Proof: Suppose that a path $(u_1, u_2, \ldots, u_i)$ has already been determined such that u_{j+1} is a descendant of u_j, for each $j = 1, 2, \ldots, i-1$. By Lemma 10.11, there exists an edge (u_i, u_{i+1}) in G such that $\sigma(u_{i+1})$ is a descendant of $\sigma(u_i)$.

Algorithm 9 CHORDLESS PATHS IN OUTERCLUSTERED GRAPHS

Require: $C(G,T)$, u, and u_1, as in Corollary 10.1.
Ensure: A path $(u_1, u_2, \ldots, u_k)$ such that $u_k = u$ and such that $\sigma(u_{i+1})$ is a descendant of $\sigma(u_i)$.

1: $u_j \leftarrow u_1$;
2: **while** $u_j \neq u$ **do**
3: $u_i \leftarrow u_j$;
4: $(u_i, u_{i+1}) \leftarrow$ an edge in G such that $\sigma(u_{i+1})$ is a descendant of $\sigma(u_i)$;
5: add (u_i, u_{i+1}) to the current path;
6: $u_j \leftarrow u_{i+1}$;
7: **end while**
8: $(u_1, u_2, \ldots, u_k) \leftarrow$ the path obtained by Algorithm 10 on G and the current path;
9: **return** $(u_1, u_2, \ldots, u_k)$;

Algorithm 10 REPLACE CHORDS IN PATHS

Require: A graph G and a path $(u_1, u_2, \ldots, u_k)$ in G.
Ensure: A chordless path in G whose vertices are a subset of $\{u_1, u_2, \ldots, u_k\}$ in the same order as in $(u_1, u_2, \ldots, u_k)$.
1: **while** the current path $(u_1, u_2, \ldots, u_k)$ has a chord (u_i, u_j) in G **do**
2: replace the subpath of $(u_1, u_2, \ldots, u_k)$ between u_i and u_j with edge (u_i, u_j);
3: **end while**
4: **return** the obtained path;

Repeating such an argument eventually leads to choose a vertex u_k in $\sigma(u)$. Since $\sigma(u) \neq \sigma(v), \sigma(z)$, then, by Property O3 of Definition 10.1, u is the only vertex in $\sigma(u)$. It follows that $u_k = u$. The obtained path $\mathcal{P}_u$ may have chords. Suppose that the currently considered path $\mathcal{P}_u^l = (u_1^l, u_2^l, \ldots, u_k^l)$ has a chord (u_i^l, u_j^l), with $j > i+1$ and with $\mathcal{P}_u^1 = \mathcal{P}_u$. Obtain a new path $\mathcal{P}_u^{l+1}$ by replacing the subpath $(u_i^l, u_{i+1}^l, \ldots, u_j^l)$ of $\mathcal{P}_u^l$ with the chord (u_i^l, u_j^l). Clearly, $\sigma(u_j^l)$ is a descendant of $\sigma(u_i^l)$. Further, $\mathcal{P}_u^{l+1}$ has at least one chord less than $\mathcal{P}_u^l$ which implies that, after a certain number of steps, the current path is chordless. □

A pseudo-code description of the algorithm for finding a path satisfying the requirements of Corollary 10.1 in a maximal outerclustered graph is presented

in Algorithm 9–10.

Let $C(G,T)$ be a maximal outerclustered graph and let u, v, and z be the vertices incident to $o(G)$. Suppose that: (i) $\sigma(u) \neq \sigma(v)$, $\sigma(u) \neq \sigma(z)$, and $\sigma(v) \neq \sigma(z)$; (ii) if there exists a cluster containing exactly two vertices incident to $o(G)$, then such vertices are u and v; (iii) if the smallest containing cluster of one of u and v contains the other one, then $\sigma(v)$ contains u; and (iv) G has internal vertices. The following lemma states that in a maximal outerclustered graph some edge-disjoint paths can be found satisfying certain properties of "monotonicity" with respect to the cluster hierarchy. After the lemma, it will be shown how such paths, together with their chords, split C into linearly-ordered outerclustered graphs.

Lemma 10.12 *One of the following holds:*

1. *There exist three paths $\mathcal{P}_u = (u_1, u_2, \ldots, u_U)$, $\mathcal{P}_v = (v_1, v_2, \ldots, v_V)$, and $\mathcal{P}_z = (z_1, z_2, \ldots, z_Z)$ such that (see Figs. 10.18.a–10.18.c):*

   a) *$u_U = u$, $v_V = v$, $z_Z = z$, and $u_1 = v_1 = z_1$;*

   b) *the vertices of $\mathcal{P}_u \setminus \{u_1\}$, $\mathcal{P}_v \setminus \{v_1\}$, and $\mathcal{P}_z \setminus \{z_1\}$ are distinct;*

   c) *each of paths $\mathcal{P}_u \setminus \{u_1\}$, $\mathcal{P}_v \setminus \{v_1\}$, and $\mathcal{P}_z$ has no chords;*

   d) *$\sigma(u_i)$ contains neither v nor z, for each $2 \leq i \leq U$; $\sigma(v_i)$ contains neither u nor z, for each $2 \leq i \leq V$; $\sigma(z_i)$ contains neither u nor v, for each $Z^* \leq i \leq Z$, where Z^* is an index such that $1 \leq Z^* \leq Z$;*

   e) *$\sigma(u_{i+1})$ is a descendant of $\sigma(u_i)$, for each $2 \leq i \leq U-1$; $\sigma(v_{i+1})$ is a descendant of $\sigma(v_i)$, for each $2 \leq i \leq V-1$;*

   f) *$\sigma(z_1)$ is either a cluster containing z and not containing u and v (then $Z^* = 1$ and $\sigma(z_{i+1})$ is a descendant of $\sigma(z_i)$, for each $1 \leq i \leq Z-1$), or $\sigma(u,v,z)$ (then $Z^* = 2$ and $\sigma(z_{i+1})$ is a descendant of $\sigma(z_i)$, for each $1 \leq i \leq Z-1$), or a cluster not containing z and containing u and v. In the latter case $Z^* \geq 2$, $\sigma(z_{i+1})$ is an ancestor of $\sigma(z_i)$, for each $1 \leq i \leq Z^*-2$, $\sigma(z_{i+1})$ is a descendant of $\sigma(z_i)$, for each $Z^* \leq i \leq Z-1$, and either $\sigma(z_{Z^*})$ is a descendant of $\sigma(z_{Z^*-1})$ (if $\sigma(z_{Z^*-1}) = \sigma(u,v,z)$), or $\sigma(z_{Z^*})$ is not comparable with $\sigma(z_{Z^*-1})$ (if $\sigma(z_{Z^*-1})$ contains u and v and does not contain z).*

   g) *G contains an internal face having incident vertices u_1, u_2, and v_2.*

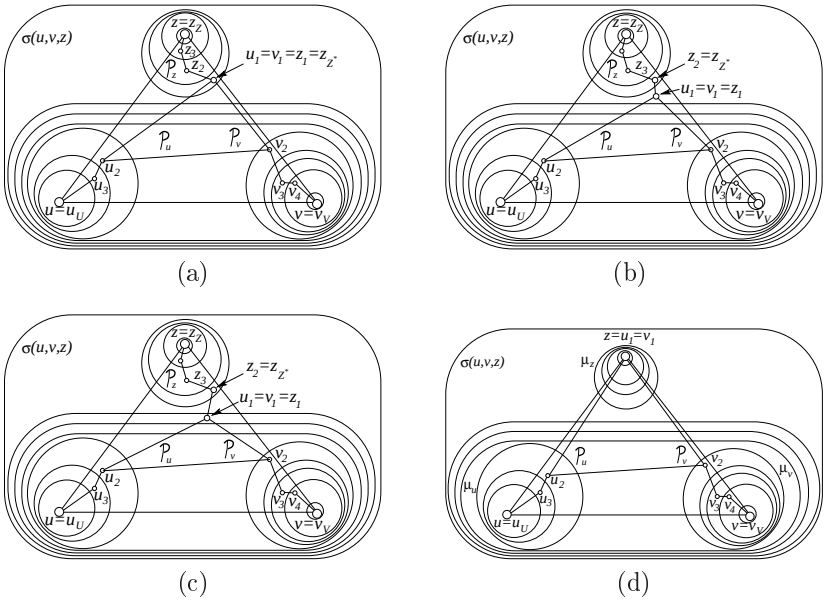


Figure 10.18: (a)–(c) Three paths satisfying Condition 1 of Lemma 10.12. In (a) $\sigma(z_1)$ is a cluster containing z and not containing u and v; in (b) $\sigma(z_1) = \sigma(u, v, z)$; in (c) $\sigma(z_1)$ is a cluster not containing z and containing u and v. (d) Two paths satisfying Condition 2 of Lemma 10.12.

2. *There exist two paths* $\mathcal{P}_u = (u_1, u_2, \ldots, u_U)$ *and* $\mathcal{P}_v = (v_1, v_2, \ldots, v_V)$ *such that (see Fig. 10.18.d):*

   a) $u_U = u$, $v_V = v$, *and* $u_1 = v_1 = z$;

   b) *the vertices of* $\mathcal{P}_u \setminus \{u_1\}$ *and* $\mathcal{P}_v \setminus \{v_1\}$ *are distinct;*

   c) *each of paths* $\mathcal{P}_u \setminus \{u_1\}$ *and* $\mathcal{P}_v \setminus \{v_1\}$ *has no chords;*

   d) $\sigma(u_i)$ *contains neither* v *nor* z, *for each* $2 \leq i \leq U$; $\sigma(v_i)$ *contains neither* u *nor* z, *for each* $2 \leq i \leq V$;

   e) $\sigma(u_{i+1})$ *is a descendant of* $\sigma(u_i)$, *for each* $2 \leq i \leq U - 1$; $\sigma(v_{i+1})$ *is a descendant of* $\sigma(v_i)$, *for each* $2 \leq i \leq V - 1$;

   f) G *contains an internal face having incident vertices* u_2, v_2, *and* z.

Proof: Consider the biggest cluster μ_u containing u and not containing v. Notice that such a cluster exists, as $\sigma(u)$ does not contain v by hypothesis. Consider the biggest cluster μ_v containing v and not containing u, if any such a cluster exists. If μ_v exists, let E' be the set of edges whose end-vertices are one in μ_u and one in μ_v. If μ_v does not exist, let E' be the set of edges incident to v whose other end-vertex is in μ_u. In both cases, there exists at least one of such edges, in fact (u, v). Order the edges in E' as they are encountered when walking on the part $l(\mu_u)$ of the border $B(\mu_u)$ of μ_u that is internal to G, starting from the intersection of (u, v) with $B(\mu_u)$. See Fig. 10.19.

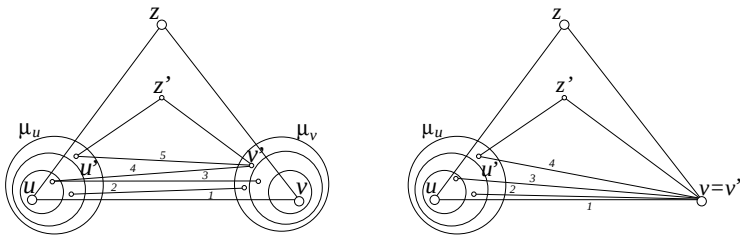


Figure 10.19: The edges in E'. The numbers on such edges indicate the order in which such edges are encountered when walking on the part of $B(\mu_u)$ internal to G, starting from the intersection of (u, v) with $B(\mu_u)$. (a) μ_v exists. (b) μ_v does not exist.

Consider the last edge (u', v') in E' and consider the internal face (u', v', z') such that (u', z') is the edge following (u', v') in the order in which the edges incident to μ_u are encountered when walking on the part of $B(\mu_u)$ that is internal to G, starting from the intersection of (u, v) with $B(\mu_u)$. Let $\mathcal{P}_u = (u_1, u_2, \ldots, u_U)$ be a path such that $u_1 = z'$, $u_2 = u'$, and $(u_2, u_3, \ldots, u_U)$ is a chordless path such that $u_U = u$ and $\sigma(u_{i+1})$ is a descendant of $\sigma(u_i)$, for $2 \leq i \leq U-1$. Such a path exists by Corollary 10.1 (notice that $\sigma(u_2)$ contains u and contains neither v nor z). Further, let $\mathcal{P}_v = (v_1, v_2, \ldots, v_V)$ be a path such that $v_1 = z'$, $v_2 = v'$, and $(v_2, v_3, \ldots, v_V)$ is a chordless path such that $v_V = v$ and $\sigma(v_{i+1})$ is a descendant of $\sigma(v_i)$, for $2 \leq i \leq V-1$. Notice that if μ_v exists, then such a path exists by Corollary 10.1 (notice that $\sigma(v_2)$ contains v and contains neither u nor z). Otherwise, $v_2 = v_V = v$.

Suppose that $z' = z$ (see Fig. 10.20). Then, G contains an internal face having incident vertices u_2, v_2, and z. By construction, paths $\mathcal{P}_u$ and $\mathcal{P}_v$ satisfy Condition 2 of the lemma.

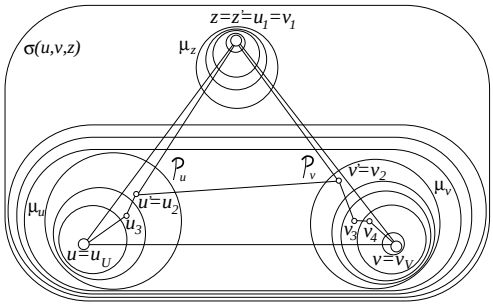


Figure 10.20: If $z' = z$, then paths $\mathcal{P}_u$ and $\mathcal{P}_v$ satisfy Condition 2 of Lemma 10.12.

Suppose that $z \neq z'$. Vertex z' belongs neither to μ_u nor to μ_v, otherwise edge (z', v') or edge (z', u') would follow (u', v') in E', a contradiction. Hence, $\sigma(z')$ is either a cluster containing z and not containing u and v, or is $\sigma(u, v, z)$, or is a cluster containing u and v and not containing z.

- Suppose that $\sigma(z')$ contains z and contains neither u nor v (see Fig. 10.21). Let $\mathcal{P}_z = (z_1, z_2, \ldots, z_Z)$ be a chordless path such that $z_1 = z'$, $z_Z = z$, and $\sigma(z_{i+1})$ is a descendant of $\sigma(z_i)$, for $2 \leq i \leq Z - 1$. Such a path exists by Corollary 10.1 (notice that $\sigma(z_1)$ contains z and contains neither u nor v). By construction, paths $\mathcal{P}_u$, $\mathcal{P}_v$, and $\mathcal{P}_z$ satisfy Condition 1 of the lemma.

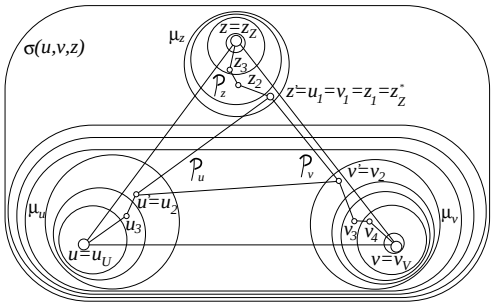


Figure 10.21: If $z' \neq z$ and $\sigma(z')$ contains z and contains neither u nor v, then paths $\mathcal{P}_u$, $\mathcal{P}_v$, and $\mathcal{P}_z$ satisfy Condition 1 of Lemma 10.12.

- Suppose that $\sigma(z') = \sigma(u, v, z)$. Refer to a c-planar embedding of C. Let μ_z be the biggest cluster containing z and containing neither u nor v.

  We claim that μ_z exists. Suppose, for a contradiction, that μ_z does not exist. Then, every cluster containing z also contains one between u and v; therefore, any such a cluster contains both u and v, by the assumptions of the lemma. Hence, $\sigma(z) = \sigma(u, v, z) = \sigma(z')$, and $\sigma(u, v, z)$ is the smallest containing cluster of at least two vertices, namely z and z'. We prove that there exists an edge between two vertices whose smallest containing cluster is $\sigma(z)$, thus contradicting the fact that C satisfies Property O3, and hence proving that μ_z exists. Consider the edges incident to z'. First, suppose that a cluster containing both u and v and not containing

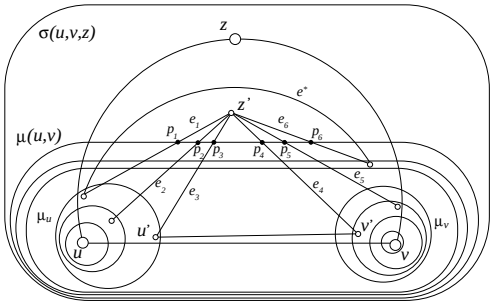


Figure 10.22: If $z' \neq z$, if $\sigma(z') = \sigma(u, v, z)$, and if a cluster containing both u and v and not containing z exists, then μ_z exists.

  z exists (see Fig. 10.22). Let $\mu_{u,v}$ be the biggest cluster containing both u and v and not containing z. Then, if no neighbor of z' has $\sigma(z')$ as smallest containing cluster, each edge e_l incident to z' intersects the boundary $B(\mu_{u,v})$ of $\mu_{u,v}$ in a point p_l belonging to the line segment $l(\mu_{u,v})$ that is the part of $B(\mu_{u,v})$ lying in the interior of $o(G)$. Order the points p_l as they are encountered when walking on $l(\mu_{u,v})$ from one endpoint to the other one. Denote by $p_1, p_2, \ldots, p_m$ such points. By the maximality of G, $m \geq 3$ and there exists a face delimited by a 3-cycle containing edges e_1 and e_m. However, the third edge e^* of such a cycle crosses twice $l(\mu_{u,v})$ and hence $B(\mu_{u,v})$, contradicting the c-planarity of the considered embedding. Second, suppose that a cluster containing both u and v and not containing z does not exist (see Fig. 10.23). Then, if μ_v exists and if no neighbor of z' has $\sigma(z')$ as smallest containing

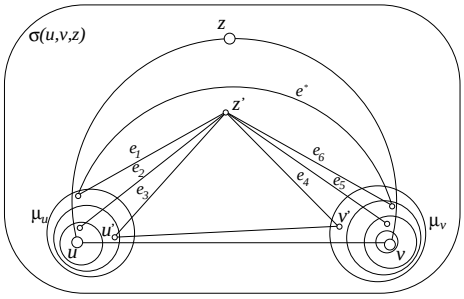


Figure 10.23: If $z' \neq z$, if $\sigma(z') = \sigma(u, v, z)$, and if a cluster containing both u and v and not containing z does not exist, then μ_z exists.

cluster, each edge e_l incident to z' is also incident either to a vertex belonging to μ_u or to a vertex belonging to μ_v; if μ_v does not exist and if no neighbor of z' has $\sigma(z')$ as smallest containing cluster, each edge e_l incident to z' is also incident to a vertex belonging to μ_u. By the c-planarity of the embedding, the order of the edges incident to z' is a sequence $(e_1, e_2, \ldots, e_i, e_{i+1}, \ldots, e_m)$ so that: (i) e_j is incident to a vertex in μ_u, for $j = 1, 2, \ldots, i$, (ii) $e_i = (z', u')$, (iii) $e_{i+1} = (z', v')$, and (iv) e_j is incident to a vertex in μ_v, for $j = i+1, i+2, \ldots, m$. Observe that, if μ_v does not exist, $e_{i+1} = (z', v)$ and $m = i + 1$. By the maximality of G, $m \geq 3$ and there exists a face delimited by a 3-cycle containing edges e_1 and e_m. However, the third edge e^* of such a cycle is an edge between a vertex in μ_u and a vertex in μ_v (if μ_v exists) or is an edge between a vertex in μ_u and v (if μ_v does not exist), contradicting the fact that (u', v') is the last edge in E'. This completes the proof of the claim that μ_z exists.

Next, we claim that there exists an edge incident to z' and to a vertex belonging to μ_z. The proof of such a claim can be done analogously to the proof that μ_z exists, namely by supposing, for a contradiction, that no edge incident to z' is incident to a vertex belonging to μ_z, and by showing that this implies that the considered embedding is not c-planar (if a cluster containing both u and v and not containing z exists), or that (u', v') is not the last edge in E' (if a cluster containing both u and v and not containing z does not exist).

Then, there exists an edge (z', z_2) incident to z' such that z_2 belongs

to μ_z. Let $(z_2, z_3, \ldots, z_Z)$ be a chordless path such that $z_Z = z$ and z_{i+1} is a descendant of z_i, for $2 \leq i \leq Z-1$, which exists by Corollary 10.1 ($\sigma(z_2)$ contains z and contains neither u nor v). Consider path $(z_1, z_2, z_3, \ldots, z_Z)$. For each chord (z_1, z_j), with $j > 2$, replace the subpath of the current path with (z_1, z_j), resulting in a chordless path $\mathcal{P}_z$. By construction, paths $\mathcal{P}_u$, $\mathcal{P}_v$, and $\mathcal{P}_z$ satisfy Condition 1 of the lemma.

- Suppose that $\sigma(z')$ contains u and v and does not contain z. In the following we show how to find a path $\mathcal{P}_z^1 = (z_1, z_2, \ldots, z_{Z^*})$ such that: (i) $z_1 = z'$, (ii) $\sigma(z_i)$ contains u and v and does not contain z, for $1 \leq i \leq Z^* - 2$, (iii) $\sigma(z_{Z^*-1})$ is either $\sigma(u, v, z)$ or a cluster containing u and v and not containing z, (iv) z_{Z^*} belongs to μ_z, and (v) $\sigma(z_{i+1})$ is an ancestor of $\sigma(z_i)$, for each $1 \leq i \leq Z^* - 2$.

  Assume that $\mathcal{P}_z^1$ has already been determined till a vertex z_i. We claim that there exists an edge incident to z_i and to a vertex z_{i+1} such that either $\sigma(z_{i+1})$ is an ancestor of $\sigma(z_i)$ containing u and v and not containing z, or $\sigma(z_{i+1}) = \sigma(u, v, z)$, or $\sigma(z_{i+1})$ contains z and does not contain u and v.

  Suppose, for a contradiction, that every edge incident to z_i is incident to a vertex belonging to μ_u, or to a vertex belonging to μ_v, or to a vertex belonging to a cluster that contains u and v, that does not contain z, and that is a descendant of $\sigma(z_i)$. Observe that vertex z_i is not adjacent to any vertex v_j such that $\sigma(v_j) = \sigma(z_i)$, by Property O3 of Definition 10.1.

  First, consider the case in which $\sigma(z_i) \neq \sigma(u, v)$, that is, there exists a child $\mu(u, v, \neg z_i)$ of $\sigma(z_i)$ in T containing u and v and not containing z. Then, each edge incident to z_i is also incident to a vertex belonging to $\mu(u, v, \neg z_i)$. Analogously as above, a contradiction can be reached by proving that there exists an edge e^* that crosses twice the boundary $B(\mu(u, v, \neg z_i))$ of $\mu(u, v, \neg z_i)$.

  Second, consider the case in which $\sigma(z_i) = \sigma(u, v)$. Then, a contradiction can be reached by proving that: (i) if all the edges incident to z_i are also incident to vertices belonging to μ_u (or if all the edges incident to z_i are also incident to vertices belonging to μ_v), then there exists an edge that crosses twice the boundary $B(\mu_u)$ of μ_u (resp. the boundary $B(\mu_v)$ of μ_v); (ii) if some edges incident to z_i are also incident to vertices in μ_u and some edges incident to z_i are also incident to vertices in μ_v, then (u', v') is not the last edge in E'.

This proves the claim, namely that there exists an edge incident to z_i and to a vertex z_{i+1} such that either $\sigma(z_{i+1})$ is an ancestor of $\sigma(z_i)$ containing u and v and not containing z, or $\sigma(z_{i+1}) = \sigma(u, v, z)$, or $\sigma(z_{i+1})$ contains z and does not contain u and v. The repetition of such an argument eventually leads to the choice of a vertex z_{Z^*} that belongs to μ_z or to the choice of a vertex z_{Z^*-1} such that $\sigma(z_{Z^*-1}) = \sigma(u, v, z)$. If z_{Z^*} belongs to μ_z, then $\mathcal{P}_z^1$ has already been determined satisfying the desired properties. If $\sigma(z_{Z^*-1}) = \sigma(u, v, z)$ then the same argument as above shows that there exists an edge (z_{Z^*-1}, z_{Z^*}) such that z_{Z^*} belongs to μ_z, thus obtaining a path $\mathcal{P}_z^1$ satisfying the desired properties.

Let $\mathcal{P}_z^2 = (z_{Z^*}, z_{Z^*+1}, \ldots, z_Z)$ be a chordless path such that $z_Z = z$, and z_{i+1} is a descendant of z_i, for $Z^* \leq i \leq Z - 1$. Such a path exists by Corollary 10.1 ($\sigma(z_{Z^*})$ contains z and contains neither u nor v).

Finally, consider the path obtained by concatenating $\mathcal{P}_z^1$ and $\mathcal{P}_z^2$. Such a path may eventually have chords. For each chord (z_i, z_j), with $j > i+1$, replace the subpath of the current path with (z_i, z_j), which results in a chordless path $\mathcal{P}_z$. By construction, paths $\mathcal{P}_u$, $\mathcal{P}_v$, and $\mathcal{P}_z$ satisfy Condition 1 of the lemma.

□

A pseudo-code description of the algorithm for finding three paths satisfying Condition 1 of Lemma 10.12 or two paths satisfying Condition 2 of Lemma 10.12 is presented in Algorithm 11.

As in Lemma 10.12, let $C(G, T)$ be a maximal outerclustered graph, let u, v, and z be the vertices incident to $o(G)$, and suppose that: (i) $\sigma(u) \neq \sigma(v)$, $\sigma(u) \neq \sigma(z)$, and $\sigma(v) \neq \sigma(z)$; (ii) if there exists a cluster containing exactly two vertices incident to $o(G)$, then such vertices are u and v; (iii) if the smallest containing cluster of one of u and v contains the other one, then $\sigma(v)$ contains u; and (iv) G has internal vertices.

Suppose that Condition 1 of Lemma 10.12 holds (see Fig. 10.24).

Denote by $C_{u,v}$, by $C_{u,z}$, and by $C_{v,z}$ the clustered graphs whose underlying graphs $G_{u,v}$, $G_{u,z}$, and $G_{v,z}$ are the subgraphs of G induced by the vertices incident to and internal to cycle $\mathcal{C}_{u,v} \equiv (u, v) \cup (\mathcal{P}_u \setminus \{u_1\}) \cup (u_2, v_2) \cup (\mathcal{P}_v \setminus \{v_1\})$, incident to and internal to cycle $\mathcal{C}_{u,z} \equiv (u, z) \cup \mathcal{P}_u \cup \mathcal{P}_z$, and incident to and internal to cycle $\mathcal{C}_{v,z} \equiv (v, z) \cup \mathcal{P}_v \cup \mathcal{P}_z$, and whose inclusion trees $T_{u,v}$, $T_{u,z}$, and $T_{v,z}$ are the subtrees of T induced by the clusters containing vertices of $G_{u,v}$, of $G_{u,z}$, and of $G_{v,z}$, respectively.

Algorithm 11 TWO OR THREE PATHS IN OUTERCLUSTERED GRAPHS

Require: $C(G,T)$, as in Lemma 10.12.
Ensure: Paths $\mathcal{P}_u$, $\mathcal{P}_v$, and $\mathcal{P}_z$ satisfying Condition 1 of Lemma 10.12, or paths $\mathcal{P}_u$ and $\mathcal{P}_v$ satisfying Condition 2 of Lemma 10.12

1: $\mu_u \leftarrow$ biggest cluster containing u and not containing v;
2: $\mu_v \leftarrow$ biggest cluster containing v and not containing u (if it exists);
3: **if** μ_v exists **then**
4: $E' \leftarrow$ set of edges whose end-vertices are one in μ_u and one in μ_v;
5: **else**
6: $E' \leftarrow$ set of edges incident to v whose other end-vertex is in μ_u;
7: **end if**
8: $(u', v') \leftarrow$ last of the edges in E' incident to $l(\mu_u)$ starting from (u, v);
9: $z' \leftarrow$ vertex forming a face with (u', v');
10: $\mathcal{P}_u \leftarrow$ edge (z', u') plus the path obtained by Algorithm 9 on C and u';
11: $\mathcal{P}_v \leftarrow$ edge (z', v') plus the path obtained by Algorithm 9 on C and v';
12: **if** $z' = z$ **then**
13: **return** $\mathcal{P}_u$ and $\mathcal{P}_v$;
14: **else**
15: **if** $\sigma(z')$ contains z and contains neither u nor v **then**
16: $\mathcal{P}_z \leftarrow$ the path obtained by Algorithm 9 on C and z';
17: **return** $\mathcal{P}_u$, $\mathcal{P}_v$, and $\mathcal{P}_z$;
18: **end if**
19: **if** $\sigma(z') = \sigma(u, v, z)$ **then**
20: $(z', z_2) \leftarrow$ edge such that z_2 is in a cluster containing z and neither u nor v;
21: $(z_2, z_3, \ldots, z_Z) \leftarrow$ path obtained by Algorithm 9 on C and z_2;
22: $\mathcal{P}_z \leftarrow$ the path obtained by Algorithm 10 on G and $(z', z_2, z_3, \ldots, z_Z)$;
23: **return** $\mathcal{P}_u$, $\mathcal{P}_v$, and $\mathcal{P}_z$;
24: **end if**
25: **if** $\sigma(z')$ contains u and v and does not contain z **then**
26: $z_j \leftarrow z'$;
27: **while** $\sigma(z_j)$ contains u and v and does not contain z **do**
28: $z_i \leftarrow z_j$; $(z_i, z_{i+1}) \leftarrow$ edge such that $\sigma(z_{i+1})$ is not a descendant of $\sigma(z_i)$;
29: add (z_i, z_{i+1}) to the current path; $z_j \leftarrow z_{i+1}$;
30: **end while**
31: **if** $\sigma(z_i) = \sigma(u, v, z)$ **then**
32: $(z_i, z_{i+1}) \leftarrow$ edge such that $\sigma(z_{i+1})$ contains z and neither u nor v;
33: add (z_i, z_{i+1}) to the current path;
34: **end if**
35: $\mathcal{P}_z^1 = (z_1, z_2, \ldots, z_{Z^*}) \leftarrow$ the current path;
36: $\mathcal{P}_z^2 = (z_{Z^*}, z_{Z^*+1}, \ldots, z_Z) \leftarrow$ path obtained by Algorithm 9 on C and z_{Z^*};
37: $\mathcal{P}_z \leftarrow$ the path obtained by Algorithm 10 on G and $\mathcal{P}_z^1$ plus $\mathcal{P}_z^2$;
38: **return** $\mathcal{P}_u$, $\mathcal{P}_v$, and $\mathcal{P}_z$;
39: **end if**
40: **end if**

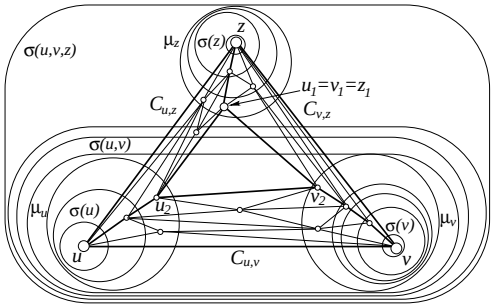


Figure 10.24: Graphs $C_{u,v}$, $C_{u,z}$, and $C_{v,z}$ when Condition 1 of Lemma 10.12 holds. Thick edges show paths $\mathcal{P}_u$, $\mathcal{P}_v$, $\mathcal{P}_z$, and edges (u,v), (u,z), (v,z), and (u_2,v_2).

Lemma 10.13 *$C_{u,v}$, $C_{u,z}$, and $C_{v,z}$ are linearly-ordered outerclustered graphs.*

Proof: We prove the statement for $C_{u,v}$, the proofs for $C_{u,z}$ and $C_{v,z}$ being analogous. Refer to a c-planar embedding of $C_{u,v}$ and to Fig. 10.25. The outer face of $G_{u,v}$ is delimited by a simple cycle $\mathcal{C}_{u,v}$. We prove that the boundary of each cluster μ containing vertices of $\mathcal{C}_{u,v}$ and not containing all the vertices of $\mathcal{C}_{u,v}$ intersects $\mathcal{C}_{u,v}$ exactly twice. By construction, each cluster containing both u and v contains all the vertices of $G_{u,v}$ and each cluster containing neither u nor v does not contain any vertex of $G_{u,v}$. Finally, each cluster μ containing u and not containing v (the arguments for each cluster μ containing v and not containing u being analogous) intersects edge (u,v) exactly once, intersects path $(\mathcal{P}_u \setminus \{u_1\}) \cup (u_2,v_2)$ exactly once, and does not intersect path $\mathcal{P}_v \setminus \{v_1\}$. It follows that μ intersects $\mathcal{C}_{u,v}$ twice. By Lemma 10.1, $C_{u,v}$ is a biconnected internally-triangulated outerclustered graph.

Consider the sequence of clusters $\Sigma = \mu_1, \mu_2, \ldots, \mu_k$ such that: (i) $\mu_1 = \sigma(u)$, (ii) μ_{i+1} is the parent of μ_i in $T_{u,v}$, for $i = 1, 2, \ldots, h-2$, (iii) μ_{h-1} is the biggest cluster containing u and not containing v, (iv) μ_h is $\sigma(u,v)$, (v) μ_{h+1} is the biggest cluster containing v and not containing u, (vi) μ_{i+1} is the only child of μ_i in $T_{u,v}$ containing v, for $i = h+1, h+2, \ldots, k-1$, and (vii) $\mu_k = \sigma(v)$. Notice that, if a cluster containing v and not containing u does not exist, then $\mu_h = \mu_k$. In the following we prove that $C_{u,v}$ is linearly-ordered according to Σ.

We prove that $C_{u,v}$ satisfies Property LO1 of Definition 10.2. Each vertex x in $G_{u,v}$ either belongs to μ_{h-1}, or to μ_{h+1}, or is such that $\sigma(x) = \sigma(u,v)$.

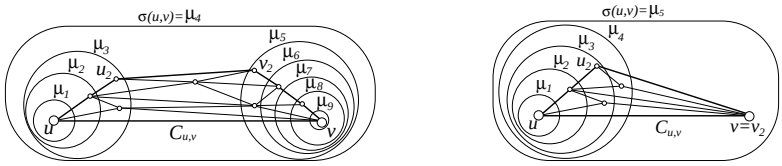


Figure 10.25: A closer look to $C_{u,v}$. Cluster sequence Σ is shown. (a) A cluster containing v and not containing u exists. (b) A cluster containing v and not containing u does not exist.

Namely, if μ_v exists, then all the vertices incident to $o(G_{u,v})$ either belong to μ_{h-1} or to μ_{h+1}, by construction; hence, all the vertices of $G_{u,v}$ belong to the smallest cluster containing both μ_{h-1} and μ_{h+1}, that is $\sigma(u,v)$, and do not belong to any cluster containing neither u nor v. If μ_v does not exist, then all the vertices incident to $o(G_{u,v})$, except for v, belong to μ_{h-1} and v belongs to $\sigma(u,v)$; hence, all the vertices of $G_{u,v}$ belong to the smallest cluster containing both μ_{h-1} and v, that is $\sigma(u,v)$, and do not belong to any cluster containing neither u nor v. Further, Σ includes all and only the clusters containing at least one of u and v, except for the clusters containing both u and v and ancestors of $\sigma(u,v)$; however, each cluster containing both u and v, and ancestor of $\sigma(u,v)$ is different from $\sigma(x)$, for any vertex x of $G_{u,v}$, since x is also contained in $\sigma(u,v)$. Since, for each internal vertex x of G, $\sigma(x)$ is a cluster containing at least one out of u and v, then $\sigma(x) = \mu_i$, for some $1 \leq i \leq k$.

We prove that $C_{u,v}$ satisfies Property LO2. Edge (u,v) and the path obtained by concatenating $\mathcal{P}_u \setminus \{u_1\}$, (u_2, v_2), and $\mathcal{P}_v \setminus \{v_1\}$ are monotone paths delimiting $o(G_{u,v})$.

We prove that $C_{u,v}$ satisfies Property LO3. By construction, μ_{i+1} is the parent of μ_i, for $i = 1, 2, \ldots, h-2$, and μ_{i+1} is a child of μ_i, for $i = h+1, h+2, \ldots, k-1$. Hence, in order to prove that $C_{u,v}$ satisfies Property LO3, it suffices to show that μ_h is the parent of μ_{h-1} and that μ_h is the parent of μ_{h+1}. By construction, μ_{h-1} (resp. μ_{h+1}) is the biggest cluster containing u and not containing v (resp. containing v and not containing u). Hence, the parent of μ_{h-1} (resp. of μ_{h+1}) is $\sigma(u,v)$, that by definition is μ_h. □

Suppose that Condition 2 of Lemma 10.12 holds (see Fig. 10.26).

Denote by $C_{u,v}$, by $C_{u,z}$, and by $C_{v,z}$ the clustered graphs whose underlying graphs $G_{u,v}$, $G_{u,z}$, and $G_{v,z}$ are the subgraphs of G induced by the vertices

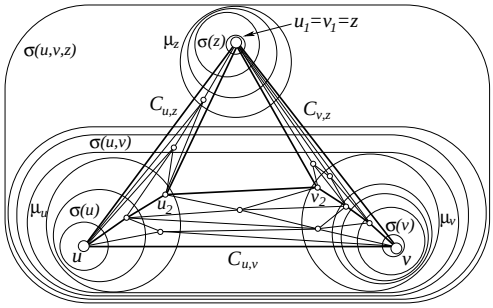


Figure 10.26: Graphs $C_{u,v}$, $C_{u,z}$, and $C_{v,z}$ when Condition 2 of Lemma 10.12 holds. Thick edges show paths $\mathcal{P}_u$, $\mathcal{P}_v$, and edges (u,v), (u,z), (v,z), and (u_2,v_2).

incident to and internal to cycle $\mathcal{C}_{u,v} \equiv (u,v) \cup (\mathcal{P}_u \setminus \{u_1\}) \cup (u_2,v_2) \cup (\mathcal{P}_v \setminus \{v_1\})$, incident to and internal to cycle $\mathcal{C}_{u,z} \equiv (u,z) \cup \mathcal{P}_u$, and incident to and internal to cycle $\mathcal{C}_{v,z} \equiv (v,z) \cup \mathcal{P}_v$, and whose inclusion trees $T_{u,v}$, $T_{u,z}$, and $T_{v,z}$ are the subtrees of T induced by the clusters containing vertices of $G_{u,v}$, of $G_{u,z}$, and of $G_{v,z}$, respectively. We have the following lemma, whose proof is analogous to the one of Lemma 10.13.

Lemma 10.14 *$C_{u,v}$, $C_{u,z}$, and $C_{v,z}$ are linearly-ordered outerclustered graphs.*

We are now ready to exhibit the main theorem of this section.

Theorem 10.2 *Let $C(G,T)$ be a maximal outerclustered graph. Then, for every triangular-convex-separated drawing $\Gamma(C_o)$ of C_o, there exists an internally-convex-separated drawing $\Gamma(C)$ of C completing $\Gamma(C_o)$.*

Proof: Let u, v, and z be the vertices incident to $o(G)$. Suppose that G has internal vertices, otherwise C_o and C are the same graph, and the statement trivially follows.

If $\sigma(u) = \sigma(v)$, then we claim that C is a linearly-ordered outerclustered graph. Refer to Fig. 10.27. Observe that C is a maximal outerclustered graph by hypothesis, hence G is biconnected and internally-triangulated.

Define the sequence of clusters $\Sigma = \mu_1, \mu_2, \ldots, \mu_k$ such that: (i) $\mu_1 = \sigma(u) = \sigma(v)$, (ii) μ_{i+1} is the parent of μ_i in T, for $i = 1, 2, \ldots, h-2$, (iii) μ_{h-1} is the biggest cluster containing u and v and not containing z, if any such a

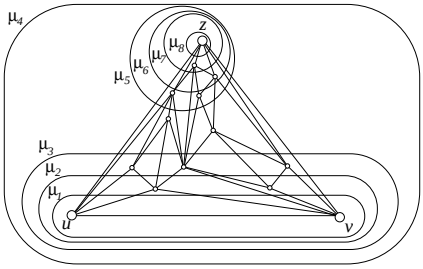


Figure 10.27: If $\sigma(u) = \sigma(v)$, then C is a linearly-ordered outerclustered graph.

cluster exists, (iv) $\mu_h = \sigma(u, v, z)$, (v) μ_{h+1} is the biggest cluster containing z and not containing u and v, if any such a cluster exists, (vi) μ_{i+1} is the only child of μ_i in T containing z, for $i = h+1, h+2, \ldots, k-1$, and (vii) $\mu_k = \sigma(z)$. We claim that C is linearly-ordered according to Σ.

We prove that C satisfies Property LO1 of Definition 10.2. By construction, Σ includes all the clusters containing at least one out of u, v, and z, with the exception of the clusters that contain all of u, v, and z and that are ancestors of $\sigma(u, v, z)$; however, each cluster containing all of u, v, and z and ancestor of $\sigma(u, v, z)$ is different from $\sigma(x)$, for any vertex x of G, since x is also contained in $\sigma(u, v, z)$. Since, for each vertex x of G, $\sigma(x)$ is a cluster containing at least one out of u, v, and z, by definition of outerclustered graph, then $\sigma(x) = \mu_i$, for some $1 \leq i \leq k$.

We prove that C satisfies Property LO2. Path (u, v, z) and edge (u, z) are monotone paths delimiting $o(G)$.

We prove that C satisfies Property LO3. By construction, μ_{i+1} is the parent of μ_i, for $i = 1, 2, \ldots, h-2$, and μ_{i+1} is a child of μ_i, for $i = h+1, h+2, \ldots, k-1$. Hence, in order to prove that C satisfies Property LO3, it suffices to show that μ_h is the parent of μ_{h-1} and that μ_h is the parent of μ_{h+1}. By construction, μ_{h-1} (resp. μ_{h+1}) is the biggest cluster containing u and v and not containing z (resp. containing z and not containing u and v). Hence, the parent of μ_{h-1} (resp. of μ_{h+1}) is $\sigma(u, v, z)$, that by definition is μ_h.

Analogously, if $\sigma(u) = \sigma(z)$ or if $\sigma(v) = \sigma(z)$, C is a linearly-ordered outerclustered graph. By Lemma 10.4, a triangular-convex-separated drawing of C_o is also a convex-separated drawing of C_o, hence in such cases the theorem directly follows from Theorem 10.1.

Now suppose that $\sigma(u) \neq \sigma(v)$, $\sigma(u) \neq \sigma(z)$, and $\sigma(v) \neq \sigma(z)$. Suppose

also that, if there exists a cluster containing exactly two vertices incident to $o(G)$, then such vertices are u and v. The cases in which such vertices are u and z, or v and z can be treated analogously. Suppose finally that if the smallest containing cluster of one of u and v contains the other one, then $\sigma(v)$ contains u. The case in which $\sigma(u)$ contains v can be treated analogously.

By Lemma 10.12, there exist three paths $\mathcal{P}_u = (u_1, u_2, \ldots, u_U)$, $\mathcal{P}_v = (v_1, v_2, \ldots, v_V)$, and $\mathcal{P}_z = (z_1, z_2, \ldots, z_Z)$ satisfying Condition 1 of Lemma 10.12, or there exist two paths $\mathcal{P}_u = (u_1, u_2, \ldots, u_U)$ and $\mathcal{P}_v = (v_1, v_2, \ldots, v_V)$ satisfying Condition 2 of Lemma 10.12.

Suppose that Condition 1 of Lemma 10.12 holds. By Lemma 10.13, $C_{u,v}(G_{u,v}, T_{u,v})$, $C_{u,z}(G_{u,z}, T_{u,z})$, and $C_{v,z}(G_{v,z}, T_{v,z})$ are linearly-ordered outerclustered graphs.

Refer to Fig. 10.28. Suppose that $U > 2$ and that $V > 2$. Consider a child $\sigma'(z_1)$ of $\sigma(z_1)$ such that: (i) if $\sigma(z_1)$ contains z and does not contain u and v, then $\sigma'(z_1)$ is the unique child of $\sigma(z_1)$; (ii) if $\sigma(z_1) = \sigma(u, v, z)$, then $\sigma'(z_1)$ is the unique child of $\sigma(z_1)$ containing z; (iii) if $\sigma(z_1)$ contains u and v and does not contain z, then $\sigma'(z_1)$ is the unique child of $\sigma(z_1)$ containing u. The existence of such clusters in each of the three cases can be proved as in the proof of Lemma 10.12. Notice that, in any case, one of the two polygonal lines obtained as intersection of the triangle representing (u, v, z) and $R(\sigma(z_1), \sigma'(z_1))$ lies on edge (u, z). Consider a point $p(z_1)$ of $int(R(\sigma(z_1), \sigma'(z_1)))$ arbitrarily close to edge (u, z). Place $u_1 = v_1 = z_1$ at $p(z_1)$.

Let $\sigma'(z_i)$ be any child of $\sigma(z_i)$ such that the border of $\sigma'(z_i)$ has intersection with (u, z), for each $2 \leq i \leq Z - 1$. Notice that such a child is unique, except for the case in which $\sigma(z_i) = \sigma(u, v, z)$, in which $\sigma(z_i)$ may have two children satisfying the required properties, namely the biggest cluster containing u and not containing z and the biggest cluster containing z and not containing u. Draw a straight-line segment $\overline{p(z_1)z}$ and place z_i at any point of the segment $int(R(\sigma(z_i), \sigma'(z_i))) \cap \overline{p(z_1)z}$, for each $2 \leq l \leq Z - 1$.

Denote by $T(u, v, p(z_1))$ the triangle with vertices u, v, and $p(z_1)$. Denote by $H(l(z, p(z_1)), u)$ (by $H(l(z, p(z_1)), v)$) the open half-plane delimited by the line through z and $p(z_1)$, and containing u (resp. containing v). Consider the unique child $\sigma'(u_2)$ of $\sigma(u_2)$. One of the two polygonal lines obtained as intersection of the triangle representing (u, v, z) and $R(\sigma(u_2), \sigma'(u_2))$ lies on edge (u, z). Consider a point $p(u_2)$ in $H(l(z, p(z_1)), u) \cap int(T(u, v, p(z_1))) \cap int(R(\sigma(u_2), \sigma'(u_2)))$ arbitrarily close to edge $(u, p(z_1))$. Place u_2 at $p(u_2)$.

Let $\sigma'(u_i)$ be the unique child of $\sigma(u_i)$, for each $3 \leq i \leq U - 1$. Draw a straight-line segment $\overline{p(u_2)u}$ and place u_i at any point of the segment $int(R(\sigma(u_i), \sigma'(u_i))) \cap \overline{p(u_2)u}$, for each $3 \leq i \leq U - 1$.

Denote by $T(p(u_2), v, p(z_1))$ the triangle having $p(u_2)$, v, and $p(z_1)$ as ver-

tices. Denote also by $H(l(u, p(u_2)), v)$ the open half-plane delimited by the line through u and $p(u_2)$, and containing v. Consider the unique child $\sigma'(v_2)$ of $\sigma(v_2)$. Consider any point $p(v_2)$ in $H(l(z, p(z_1)), v) \cap H(l(u, p(u_2)), v) \cap int(T(p(u_2), v, p(z_1))) \cap int(R(\sigma(v_2), \sigma'(v_2)))$. Observe that, since $p(z_1)$ and $p(u_2)$ are arbitrarily close to edge (u, z), both half-planes $H(l(z, p(z_1)), v)$ and $H(l(u, p(u_2)), v)$ entirely contain triangle $T(u, v, z)$, except for an arbitrarily small strip close to edge (u, z). This guarantees that $H(l(z, p(z_1)), v) \cap H(l(u, p(u_2)), v) \cap int(T(p(u_2), v, p(z_1))) \cap int(R(\sigma(v_2), \sigma'(v_2)))$ is a convex non-empty region. Then, place v_2 at $p(v_2)$.

Let $\sigma'(v_i)$ be the unique child of $\sigma(v_i)$, for each $3 \leq i \leq V-1$. Draw a straight-line segment $\overline{p(v_2)v}$ and place v_i at any point of $int(R(\sigma(v_i), \sigma'(v_i))) \cap \overline{p(v_2)v}$, for each $3 \leq i \leq V-1$. Straightforward modifications make the described algorithm work also for the cases in which $U = 2$ and/or $V = 2$. Such modifications are described in Algorithm 13 below.

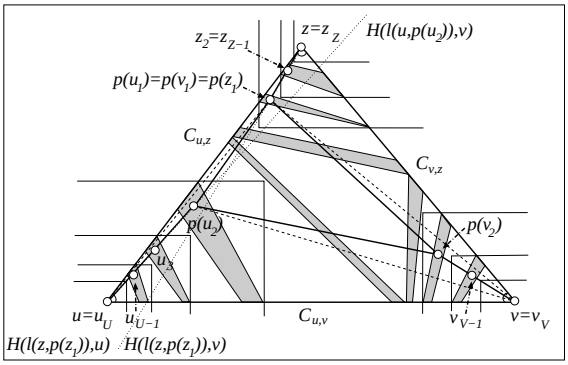


Figure 10.28: Construction of drawings $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$, $\Gamma(\mathcal{C}_{u,z}, T_{u,z})$, and $\Gamma(\mathcal{C}_{v,z}, T_{v,z})$ when Condition 1 of Lemma 10.12 holds.

Denote by $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$, by $\Gamma(\mathcal{C}_{u,z}, T_{u,z})$, and by $\Gamma(\mathcal{C}_{v,z}, T_{v,z})$ the constructed drawings of $(\mathcal{C}_{u,v}, T_{u,v})$, of $(\mathcal{C}_{u,z}, T_{u,z})$, and of $(\mathcal{C}_{v,z}, T_{v,z})$, respectively. We have the following:

Lemma 10.15 *$\Gamma(\mathcal{C}_{u,v}, T_{u,v})$, $\Gamma(\mathcal{C}_{u,z}, T_{u,z})$, and $\Gamma(\mathcal{C}_{v,z}, T_{v,z})$ are convex-separated drawings of the outer faces of $C_{u,v}(G_{u,v}, T_{u,v})$, $C_{u,z}(G_{u,z}, T_{u,z})$, and $C_{v,z}(G_{v,z}, T_{v,z})$, respectively.*

Proof: We prove the statement for $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$, the proof for $\Gamma(\mathcal{C}_{u,z}, T_{u,z})$, and $\Gamma(\mathcal{C}_{v,z}, T_{v,z})$ being analogous. Denote by P, $P_{u,v}$, $P_{u,z}$, and $P_{v,z}$ the polygons representing cycles $\mathcal{C}$, $\mathcal{C}_{u,v}$, $\mathcal{C}_{u,z}$, and $\mathcal{C}_{v,z}$, respectively. The drawing is straight-line and rectangular by construction. The absence of edge crossings easily descends from the construction. The absence of region-region crossings descends from the fact that no cluster is drawn by the algorithm.

We prove that $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$ has no edge-region crossings. Suppose, for a contradiction, that there is an edge-region crossing between an edge e and a cluster ν. If both endvertices of e belong to ν then, by the convexity of ν, e is internal to ν; if one endvertex of e belongs to ν then, by the convexity of ν, e crosses ν exactly once; hence, it can be assumed that both the endvertices of e do not belong to ν. Any cluster containing both u and v contains all the vertices of $G_{u,v}$, hence it contains all the drawing of $\mathcal{C}_{u,v}$ and does not cross e. Hence, it can be assumed that ν contains u and does not contain v, or vice versa. Suppose that ν contains u and does not contain v, the other case being analogous. Consider the parent μ of ν in $T_{u,v}$. Such a parent exists since otherwise ν would be the root of T, contradicting the fact that ν does not contain v. By definition of triangular-convex-separated drawing, there exists a convex region $R(\mu, \nu)$ with the properties described in Definition 10.4; one of the sides of such a region separates ν from the rest of the drawing, thus avoiding an edge-region crossing between e and ν. More precisely, since $C(G, T)$ is an outerclustered graph, ν has exactly two incident edges $e_1(\nu)$ and $e_2(\nu)$ incident to $o(G)$. Denote by $u(e_1(\nu))$ and $u(e_2(\nu))$ the endvertices of $e_1(\nu)$ and $e_2(\nu)$ belonging to ν. Denote by $p(l_1)$ the endpoint of $l_1(\mu, \nu)$ that lies on $e_1(\nu)$ (if both endpoints of $l_1(\mu, \nu)$ lie on $e_1(\nu)$, then $p(l_1)$ is the one that is closer to $u(e_1(\nu))$). Note that an endpoint of $l_1(\mu, \nu)$ lying on $e_1(\nu)$ exists as Γ satisfies Property CS3. Analogously define $p(l_2)$. Then, segment $\overline{p(l_1)p(l_2)}$ splits P into two disjoint convex polygons P' and P'', where P' contains all and only the vertices belonging to ν and P'' contains all and only the vertices not belonging to ν, as $\Gamma(C_o)$ satisfies Property CS3. By convexity, e is internal to P'', while the part of ν inside P is internal to P'. Hence, e does not cross ν.

We prove that $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$ satisfies Property CS1 of Definition 10.3. The angles $\widehat{u_2uv}$ and $\widehat{v_2vu}$ incident to u and v inside $P_{u,v}$ are strictly less than 180°, since they are respectively less than angles $\widehat{zuv}$ and $\widehat{zvu}$, that are angles of P, which is a triangle. By construction, v_2 is contained inside triangle $T(p(u_2), v, p(z_1))$, hence $\widehat{u_2v_2v}$ is the angle of a triangle having u_2, v_2, and v as vertices, hence it is less than 180°. Finally, angle $\widehat{uu_2v_2}$ is less than 180°, since by construction v_2 is placed in the half-plane $H(l(u, p(u_2)), v)$ delimited by the line through u and $p(u_2)$, and containing v.

We prove that $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$ satisfies Property CS2. Observe that $\sigma(u)$ and $\sigma(v)$ are the first and the last cluster in Σ, respectively, and that the angles $\widehat{u_2uv}$ and $\widehat{v_2vu}$ incident to u and v inside $P_{u,v}$ are strictly less than 180°, as proved above.

We prove that $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$ satisfies Property CS3. The existence of regions $R(\mu, \nu)$ inside $P_{u,v}$ descends from the existence of regions $R(\mu, \nu)$ inside P, where ν is any child of μ in $T_{u,v}$. Namely, the drawn edges cut each convex region $R(\mu, \nu)$ into two or three convex regions. Such regions satisfy the properties that have to be satisfied by $R(\mu, \nu)$ inside $P_{u,v}$, as can be easily deduced from the fact that the same properties are satisfied by $R(\mu, \nu)$ inside P. □

Graphs $C_{u,v}$, $C_{u,z}$, and $C_{v,z}$ are, in general, not triconnected since there could exist chords: (i) in $\mathcal{C}_{u,v}$ between any vertex in $\mathcal{P}_u \setminus \{u_1\}$ and any vertex in $\mathcal{P}_v \setminus \{v_1\}$; (ii) in $\mathcal{C}_{u,z}$ between vertex u_1 and any vertex in $\mathcal{P}_u \setminus \{u_1\}$, and between any vertex in $\mathcal{P}_u \setminus \{u_1\}$ and any vertex in $\mathcal{P}_z \setminus \{z_1\}$; (iii) in $\mathcal{C}_{v,z}$ between vertex v_1 and any vertex in $\mathcal{P}_v \setminus \{v_1\}$, and between any vertex in $\mathcal{P}_v \setminus \{v_1\}$ and any vertex in $\mathcal{P}_z \setminus \{z_1\}$. By Lemma 10.2, each of such chords splits a linearly-ordered outerclustered graph into two smaller linearly-ordered outerclustered graphs. Further, by construction the endvertices of each of such chords are not collinear with any other vertex of the cycle. Hence, by Lemma 10.3, inserting the chords as straight-line segments into drawings $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$, $\Gamma(\mathcal{C}_{u,z}, T_{u,z})$, and $\Gamma(\mathcal{C}_{v,z}, T_{v,z})$, that are convex-separated by Lemma 10.15, splits them into convex-separated drawings. When all the chords have been added, the underlying graphs of the resulting clustered graphs are all triconnected and internally-triangulated. Hence, Theorem 10.1 applies and an internally-convex-separated drawing of each of such linearly-ordered outerclustered graphs can be constructed inside the corresponding outer face, thus obtaining an internally-convex-separated drawing of C.

Now suppose that Condition 2 of Lemma 10.12 holds.

By Lemma 10.14, $C_{u,v}(G_{u,v}, T_{u,v})$, $C_{u,z}(G_{u,z}, T_{u,z})$, and $C_{v,z}(G_{v,z}, T_{v,z})$ are linearly-ordered outerclustered graphs.

Refer to Fig. 10.29. Suppose that $U > 2$ and that $V > 2$. Consider the unique child $\sigma'(u_i)$ of $\sigma(u_i)$, for each $2 \leq i \leq U-1$. Notice that one of the two polygonal lines obtained as intersection of the triangle representing (u, v, z) and $R(\sigma(u_2), \sigma'(u_2))$ lies on edge (u, z). Consider a point $p(u_2)$ of $int(R(\sigma(u_2), \sigma'(u_2)))$ arbitrarily close to (u, z). Place u_2 at $p(u_2)$. Draw a straight-line segment $\overline{p(u_2)u}$ and place u_i at any point of the segment $int(R(\sigma(u_i), \sigma'(u_i))) \cap \overline{p(u_2)u}$, for each $3 \leq i \leq U-1$.

Denote by $T(p(u_2), v, z)$ the triangle having $p(u_2)$, v, and z as vertices. De-

note also by $H(l(u, p(u_2)), v)$ the open half-plane delimited by the line through u and $p(u_2)$, and containing v.

Let $\sigma'(v_i)$ be the unique child of $\sigma(v_i)$, for each $2 \leq i \leq V-1$. Consider any point $p(v_2)$ in $H(l(u, p(u_2)), v) \cap int(T(p(u_2), v, z)) \cap int(R(\sigma(v_2), \sigma'(v_2)))$ and place v_2 at $p(v_2)$. Draw a straight-line segment $\overline{p(v_2)v}$ and place v_i at any point of the segment $int(R(\sigma(v_i), \sigma'(v_i))) \cap \overline{p(v_2)v}$, for each $3 \leq i \leq V-1$. Straightforward modifications make the described algorithm work also for the cases in which $U = 2$ or $V = 2$ (notice that $U = 2$ and $V = 2$ do not hold simultaneously when Condition 2 of Lemma 10.12 holds, otherwise G would not have internal vertices). Such modifications are described in Algorithm 14 below.

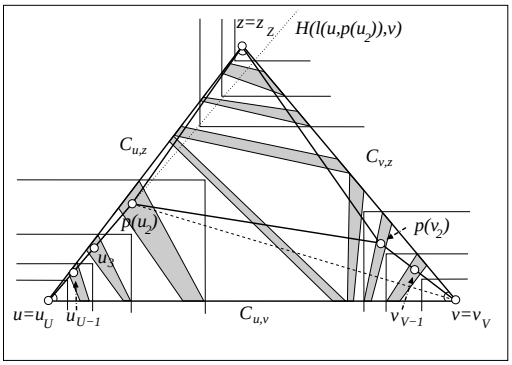


Figure 10.29: Construction of drawings $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$, $\Gamma(\mathcal{C}_{u,z}, T_{u,z})$, and $\Gamma(\mathcal{C}_{v,z}, T_{v,z})$ when Condition 2 of Lemma 10.12 holds.

Denote by $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$, by $\Gamma(\mathcal{C}_{u,z}, T_{u,z})$, and by $\Gamma(\mathcal{C}_{v,z}, T_{v,z})$ the constructed drawings of $(\mathcal{C}_{u,v}, T_{u,v})$, of $(\mathcal{C}_{u,z}, T_{u,z})$, and of $(\mathcal{C}_{v,z}, T_{v,z})$, respectively. We have the following lemma, whose proof is analogous to the proof of Lemma 10.15.

Lemma 10.16 *$\Gamma(\mathcal{C}_{u,v}, T_{u,v})$, $\Gamma(\mathcal{C}_{u,z}, T_{u,z})$, and $\Gamma(\mathcal{C}_{v,z}, T_{v,z})$ are convex-separated drawings of the outer faces of $C_{u,v}(G_{u,v}, T_{u,v})$, $C_{u,z}(G_{u,z}, T_{u,z})$, and $C_{v,z}(G_{v,z}, T_{v,z})$, respectively.*

Graphs $C_{u,v}$, $C_{u,z}$, and $C_{v,z}$ are, in general, not triconnected, since there could exist chords: (i) in $\mathcal{C}_{u,v}$ between any vertex in $\mathcal{P}_u \setminus \{u_1\}$ and any vertex in $\mathcal{P}_v \setminus \{v_1\}$; (ii) in $\mathcal{C}_{u,z}$ between vertex u_1 and any vertex in $\mathcal{P}_u \setminus \{u_1\}$; (iii) in $\mathcal{C}_{v,z}$ between vertex v_1 and any vertex in $\mathcal{P}_v \setminus \{v_1\}$. By Lemma 10.2, each

of such chords splits a linearly-ordered outerclustered graph into two smaller linearly-ordered outerclustered graphs. Further, by construction, the endvertices of each of such chords are not collinear with any other vertex of the cycle. Hence, by Lemma 10.3, inserting the chords as straight-line segments into drawings $\Gamma(\mathcal{C}_{u,v}, T_{u,v})$, $\Gamma(\mathcal{C}_{u,z}, T_{u,z})$, and $\Gamma(\mathcal{C}_{v,z}, T_{v,z})$, that are convex-separated by Lemma 10.16, splits them into convex-separated drawings. When all chords have been added, the underlying graphs of the resulting clustered graphs are all triconnected and internally-triangulated. Hence, Theorem 10.1 applies and an internally-convex-separated drawing of each of such linearly-ordered outerclustered graphs can be constructed inside the corresponding outer face, thus obtaining an internally-convex-separated drawing of C. □

A pseudo-code description of the algorithm for drawing a maximal outerclustered graph C (supposing that, if there exists a cluster containing exactly two vertices incident to $o(G)$, then such vertices are u and v, and that, if the smallest containing cluster of one of u and v contains the other one, then $\sigma(v)$ contains u) is presented in Algorithms 12–15.

10.5 How to Draw Clustered Graphs

In this section we prove that every clustered graph $C(G,T)$ admits an internally-convex-separated drawing $\Gamma(C)$ completing an arbitrary triangular-convex-separated drawing $\Gamma(C_o)$ of C_o. The result is achieved by means of an inductive algorithm, where the induction is on the number of vertices plus the number of clusters of C. In the base case, C is an outerclustered graph and the statement follows from Theorem 10.2. Consider any maximal clustered graph $C(G,T)$.

Case 1: There exists a minimal cluster containing exactly one internal vertex and no external vertex. Refer to Fig. 10.30. Let μ be a minimal cluster containing exactly one vertex v internal to G and containing no vertex incident to $o(G)$. Remove μ from T obtaining a clustered graph $C'(G,T')$. Observe that C_o and C'_o are the same graph, since μ does not contain vertices incident to $o(G)$. The number of vertices plus the number of clusters of C' is one less than the number of vertices plus the number of clusters of C. Hence, the inductive hypothesis applies and, for an arbitrary triangular-convex-separated drawing $\Gamma(C_o)$, there exists an internally-convex-separated drawing $\Gamma(C')$ of C' completing $\Gamma(C_o)$. In $\Gamma(C')$ a small disk D can be drawn with the following properties: it is centered at v, it does not intersect the boundary of any cluster, it does not contain any vertex of G diff erent from v, and it has intersection only with the edges incident to v. For each edge e_i

Algorithm 12 DRAWING OUTERCLUSTERED GRAPHS

Require: $C(G,T)$ and $\Gamma(C_o)$.
Ensure: An internally-convex-separated drawing $\Gamma(C)$ of C completing $\Gamma(C_o)$.

1: **if** G has no internal vertex **then**
2: $\Gamma(C) \leftarrow \Gamma(C_o)$;
3: **else**
4: **if** $\sigma(u) = \sigma(v)$ or $\sigma(u) = \sigma(z)$ or $\sigma(v) = \sigma(z)$ **then**
5: $\Gamma(C) \leftarrow$ the drawing obtained by applying Algorithm 8 on C and $\Gamma(C_o)$;
6: **else**
7: apply Algorithm 11 on C, obtaining either three paths $\mathcal{P}_u$, $\mathcal{P}_v$, and $\mathcal{P}_z$, or two paths $\mathcal{P}_u$ and $\mathcal{P}_v$;
8: **for** $2 \leq i \leq U-1$ **do**
9: $\sigma'(u_i) \leftarrow$ the unique child of $\sigma(u_i)$;
10: **end for**
11: **for** $2 \leq i \leq V-1$ **do**
12: $\sigma'(v_i) \leftarrow$ the unique child of $\sigma(v_i)$;
13: **end for**
14: **if** Algorithm 11 on C returns three paths $\mathcal{P}_u = (u_1, \ldots, u_U)$, $\mathcal{P}_v = (v_1, \ldots, v_V)$, and $\mathcal{P}_z = (z_1, z_2, \ldots, z_Z)$ **then**
15: **for** $1 \leq i \leq Z-1$ **do**
16: **if** $U > 2$ **then**
17: $\sigma'(z_i) \leftarrow$ any child of $\sigma(z_i)$ whose border has intersection with (u, z);
18: **else**
19: $\sigma'(z_i) \leftarrow$ any child of $\sigma(z_i)$ whose border has intersection with (v, z);
20: **end if**
21: **end for**
22: $\Gamma(C) \leftarrow$ the drawing obtained by applying Algorithm 13 on C, $\mathcal{P}_u$, $\sigma'(u_2), \ldots, \sigma'(u_{U-1})$, $\mathcal{P}_v$, $\sigma'(v_2), \ldots, \sigma'(v_{V-1})$, $\mathcal{P}_z$, $\sigma'(z_1), \ldots, \sigma'(z_{Z-1})$, and $\Gamma(C_o)$;
23: **end if**
24: **if** Algorithm 11 on C returns two paths $\mathcal{P}_u = (u_1, \ldots, u_U)$ and $\mathcal{P}_v = (v_1, \ldots, v_V)$ **then**
25: $\Gamma(C) \leftarrow$ the drawing obtained by applying Algorithm 14 on C, $\mathcal{P}_u$, $\sigma'(u_2), \ldots, \sigma'(u_{U-1})$, $\mathcal{P}_v$, $\sigma'(v_2), \ldots, \sigma'(v_{V-1})$, and $\Gamma(C_o)$;
26: **end if**
27: **end if**
28: **end if**
29: **return** $\Gamma(C)$;

Algorithm 13 DRAWING OUTERCLUSTERED GRAPHS WITH THREE PATHS

Require: C, $\mathcal{P}_u = (u_1, \ldots, u_U)$, $\sigma'(u_2), \ldots, \sigma'(u_{U-1})$, $\mathcal{P}_v = (v_1, \ldots, v_V)$, $\sigma'(v_2), \ldots, \sigma'(v_{V-1})$, $\mathcal{P}_z = (z_1, z_2, \ldots, z_Z)$, $\sigma'(z_1), \ldots, \sigma'(z_{Z-1})$, and $\Gamma(C_o)$;
Ensure: An internally-convex-separated drawing $\Gamma(C)$ of C completing $\Gamma(C_o)$.

1: **if** $U > 2$ **then**
2: $p(z_1) \leftarrow$ point of $int(R(\sigma(z_1), \sigma'(z_1)))$ close to (u, z); place z_1 at $p(z_1)$;
3: $H(l(z, p(z_1)), u), H(l(z, p(z_1)), v) \leftarrow$ the open half-planes delimited by the line through z and $p(z_1)$, and containing u and v, respectively;
4: $p(u_2) \leftarrow$ point of $H(l(z, p(z_1)), u) \cap int(T(u, v, p(z_1))) \cap int(R(\sigma(u_2), \sigma'(u_2)))$ close to $(u, p(z_1))$; place u_2 at $p(u_2)$;
5: apply Algorithm 15 on $\mathcal{P}_u \setminus \{u_1\}$, $R(\sigma(u_3), \sigma'(u_3))$, $R(\sigma(u_4), \sigma'(u_4))$, $\ldots$, $R(\sigma(u_{U-1}), \sigma'(u_{U-1}))$, and $p(u_2)u$;
6: **if** $V > 2$ **then**
7: $H(l(u, p(u_2)), v) \leftarrow$ the open half-plane delimited by the line through u and $p(u_2)$ and containing v;
8: $p(v_2) \leftarrow$ point of $H(l(z, p(z_1)), v) \cap H(l(u, p(u_2)), v) \cap int(T(p(u_2), v, p(z_1))) \cap int(R(\sigma(v_2), \sigma'(v_2)))$; place v_2 at $p(v_2)$;
9: apply Algorithm 15 on $\mathcal{P}_v \setminus \{v_1\}$, $R(\sigma(v_3), \sigma'(v_3))$, $R(\sigma(v_4), \sigma'(v_4))$, $\ldots$, $R(\sigma(v_{V-1}), \sigma'(v_{V-1}))$, and $p(v_2)v$;
10: **end if**
11: **else**
12: **if** $V > 2$ **then**
13: $p(z_1) \leftarrow$ point of $int(R(\sigma(z_1), \sigma'(z_1)))$ close to (v, z); place z_1 at $p(z_1)$;
14: $H(l(z, p(z_1)), v) \leftarrow$ the open half-plane delimited by the line through z and $p(z_1)$, and containing v;
15: $p(v_2) \leftarrow$ point of $H(l(z, p(z_1)), v) \cap int(T(u, v, p(z_1))) \cap int(R(\sigma(v_2), \sigma'(v_2)))$ close to $(v, p(z_1))$; place v_2 at $p(v_2)$;
16: apply Algorithm 15 on $\mathcal{P}_v \setminus \{v_1\}$, $R(\sigma(v_3), \sigma'(v_3))$, $R(\sigma(v_4), \sigma'(v_4))$, $\ldots$, $R(\sigma(v_{V-1}), \sigma'(v_{V-1}))$, and $p(v_2)v$;
17: **end if**
18: **end if**
19: **if** $U = 2$ and $V = 2$ **then**
20: $p(z_1) \leftarrow$ point of $int(R(\sigma(z_1), \sigma'(z_1)))$; place z_1 at $p(z_1)$;
21: **end if**
22: apply Algorithm 15 on $\mathcal{P}_z$, $R(\sigma(z_2), \sigma'(z_2))$, $R(\sigma(z_3), \sigma'(z_3))$, $\ldots$, $R(\sigma(z_{Z-1}), \sigma'(z_{Z-1}))$, and $p(z_1)z$;
23: draw each edge of G between two vertices belonging to $\mathcal{P}_u$, to $\mathcal{P}_v$, or to $\mathcal{P}_z$ as a straight-line segment;
24: apply Algorithm 8 on each resulting linearly-ordered outerclustered graph completing a drawing of the corresponding outer face;
25: $\Gamma(C) \leftarrow$ the resulting internally-convex-separated drawing of C;
26: **return** $\Gamma(C)$;

Algorithm 14 DRAWING OUTERCLUSTERED GRAPHS WITH TWO PATHS

Require: C, $\mathcal{P}_u = (u_1, \ldots, u_U)$, $\sigma'(u_2), \ldots, \sigma'(u_{U-1})$, $\mathcal{P}_v = (v_1, \ldots, v_V)$, $\sigma'(v_2), \ldots, \sigma'(v_{V-1})$, and $\Gamma(C_o)$;
Ensure: An internally-convex-separated drawing $\Gamma(C)$ of C completing $\Gamma(C_o)$.

1: **if** $U > 2$ **then**
2: $p(u_2) \leftarrow$ point of $int(R(\sigma(u_2), \sigma'(u_2)))$ close to (u, z); place u_2 at $p(u_2)$;
3: apply Algorithm 15 on $\mathcal{P}_u \setminus \{u_1\}$, $R(\sigma(u_3), \sigma'(u_3))$, $R(\sigma(u_4), \sigma'(u_4))$, ..., $R(\sigma(u_{U-1}), \sigma'(u_{U-1}))$, and $\overline{p(u_2)u}$;
4: **if** $V > 2$ **then**
5: $H(l(u, p(u_2)), v) \leftarrow$ the open half-plane delimited by the line through u and $p(u_2)$, and containing v;
6: $p(v_2) \leftarrow$ point of $H(l(u, p(u_2)), v) \cap int(T(p(u_2), v, z)) \cap int(R(\sigma(v_2), \sigma'(v_2)))$; place v_2 at $p(v_2)$;
7: apply Algorithm 15 on $\mathcal{P}_v \setminus \{v_1\}$, $R(\sigma(v_3), \sigma'(v_3))$, $R(\sigma(v_4), \sigma'(v_4))$, ..., $R(\sigma(v_{V-1}), \sigma'(v_{V-1}))$, and $\overline{p(v_2)v}$;
8: **end if**
9: **else**
10: $p(v_2) \leftarrow$ point of $int(R(\sigma(v_2), \sigma'(v_2)))$; place v_2 at $p(v_2)$;
11: apply Algorithm 15 on $\mathcal{P}_v \setminus \{v_1\}$, $R(\sigma(v_3), \sigma'(v_3))$, $R(\sigma(v_4), \sigma'(v_4))$, ..., $R(\sigma(v_{V-1}), \sigma'(v_{V-1}))$, and $\overline{p(v_2)v}$;
12: **end if**
13: draw each edge of G between two vertices belonging to $\mathcal{P}_u$ or to $\mathcal{P}_v$ as a straight-line segment;
14: apply Algorithm 8 on each resulting linearly-ordered outerclustered graph completing a drawing of the corresponding outer face;
15: $\Gamma(C) \leftarrow$ the resulting internally-convex-separated drawing of C;
16: **return** $\Gamma(C)$;

Algorithm 15 DRAWING PATH IN OUTERCLUSTERED GRAPHS

Require: $\mathcal{P}_x = (x_1, x_2, \ldots, x_X)$, $R(\sigma(x_2), \sigma'(x_2)))$, $R(\sigma(x_3), \sigma'(x_3)))$, ..., $R(\sigma(x_{X-1}), \sigma'(x_{X-1})))$, and $\overline{p(x_1)x_X}$.

1: **for** $2 \leq i \leq X - 1$ **do**
2: place x_i at any point of $int(R(\sigma(x_i), \sigma'(x_i))) \cap \overline{p(x_1)x_X}$;
3: **end for**

incident to v, choose two points p_i^1 and p_i^2 inside D, where p_i^1 is closer to v than p_i^2. For each two edges e_i and e_{i+1} consecutively incident to v denote by f_i the face of G incident to edges e_i and e_{i+1}, and denote by $R(\mu, i)$ the quadrilateral having p_i^1, p_i^2, p_{i+1}^1, and p_{i+1}^2 as vertices. Finally, insert a drawing of μ in $\Gamma(C')$ as a rectangle containing v and contained inside the polygon $(p_1^1, p_2^1, \ldots, p_k^1, p_1^1)$, thus obtaining a drawing $\Gamma(C)$.

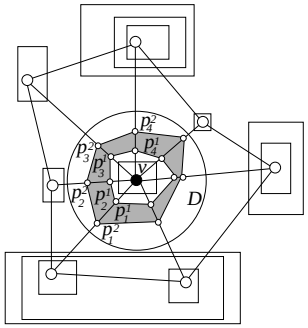


Figure 10.30: Illustration for Case 1 of the algorithm for drawing clustered graphs.

Lemma 10.17 *$\Gamma(C)$ is an internally-convex-separated drawing of C.*

Proof: $\Gamma(C)$ has no edge crossing, by induction. $\Gamma(C)$ has no edge-region crossing since any cluster different from μ has no crossing with any edge by induction, and since μ does not intersect any edge not incident to v, because it completely lies inside D. $\Gamma(C)$ has no region-region crossing since the boundaries of any two clusters different from μ have no crossing by induction, and since the boundary of μ does not intersect the boundary of any cluster, because it completely lies inside D. The drawing is straight-line and rectangular by construction. Further, every internal face of G not incident to v is triangular-convex-separated by induction. Finally, for each face f_i, region $R(\nu, \mu) \equiv R(\mu, i)$, where ν is the parent of μ in T, satisfies Property TCS1 of a triangular-convex-separated drawing, due to the fact that such a region is completely contained inside D, and that D is completely contained inside ν. □

Case 2: There exists a separating 3-cycle. Suppose that G contains a separating 3-cycle (u', v', z'). Let $C^1(G^1, T^1)$ be the clustered graph defined as follows. G^1 is the subgraph of G induced by all the vertices external to

(u', v', z'), by u', by v', and by z'. T^1 is the subtree of T whose clusters contain at least one vertex of G^1. Observe that C_o and C_o^1 are the same graph. Further, let $C^2(G^2, T^2)$ be the clustered graph defined as follows. G^2 is the subgraph of G induced by all the vertices internal to (u', v', z'), by u', by v', and by z'. T^2 is the subtree of T whose clusters contain at least one vertex of G^2. Since (u', v', z') is a separating 3-cycle, the number of vertices plus the number of clusters of each of C^1 and C^2 is strictly less than the number of vertices plus the number of clusters of C. Hence, the inductive hypothesis applies and, for an arbitrary triangular-convex-separated drawi ng $\Gamma(C_o)$, there exists an internally-convex-separated drawing $\Gamma(C^1)$ of C^1 completing $\Gamma(C_o)$. Cycle (u', v', z') is a face f of G^1. By definition of internally-convex-separated drawing of a graph, the drawing $\Gamma(C_f)$ of C_f in $\Gamma(C^1)$ is a triangular-convex-separated drawing. Observe that C_f and C_o^2 are the same graph. Hence, the inductive hypothesis applies again and an internally-convex-separated drawing $\Gamma(C^2)$ can be constructed completing $\Gamma(C_o^2)$. Plugging $\Gamma(C^2)$ into $\Gamma(C^1)$ provides a drawing $\Gamma(C)$ of C.

Lemma 10.18 *$\Gamma(C)$ is an internally-convex-separated drawing of C.*

Proof: $\Gamma(C)$ has no edge crossing. Namely, any edge belonging to G^1 (resp. to G^2) does not cross any edge belonging to G^1 (resp. to G^2) by induction. Further, any edge belonging to G^1 and not belonging to G^2 does not cross any edge belonging to G^2 and not belonging to G^1 since such edges are separated by cycle (u', v', z'). $\Gamma(C)$ has no edge-region crossing. Namely, any edge belonging to G^1 (resp. to G^2) does not cross the boundary of any cluster of T^1 (resp. of T^2) by induction. Further, any edge belonging to G^1 (resp. to G_2) and not belonging to G^2 (resp. to G^1) does not cross the boundary of any cluster belonging to T^2 (resp. to T^1) and not belonging to T^1 (resp. to T^2), since such an edge and such a cluster are separated by cycle (u', v', z'). $\Gamma(C)$ has no region-region crossing. Namely, the boundary of any cluster belonging to T^1 (resp. to T^2) does not cross the boundary of any cluster belonging to T^1 (resp. to T^2) by induction. Further, the boundary of any cluster belonging to T^1 (resp. to T^2) and not belonging to T^2 (resp. to T^1) does not cross the boundary of any cluster belonging to T^2 (resp. to T^1) and not belonging to T^1 (resp. to T^2), since such clusters are separated by cycle (u', v', z'). $\Gamma(C)$ is straight-line and rectangular by construction. Further, the drawing of any internal face f of G is triangular-convex-separated since it is triangular-convex-separated in $\Gamma(C^1)$ (if f is also a face of G^1) or in $\Gamma(C^2)$ (if f is also a face of G^2). □

Case 3: There exists no separating 3-cycle and there exists an edge (u', v') not incident to $o(G)$ such that $\sigma(u') = \sigma(v')$. Refer to Fig. 10.31. Suppose that G contains an edge (u', v') with $\sigma(u') = \sigma(v')$ such that u' is internal and suppose that there exists no separating 3-cycle containing edge (u', v'). Since G is internally-triangulated, u' and v' have exactly two common neighbors z'_1 and z'_2. Contract edge (u', v') to a vertex w', that is, replace vertices u' and v' with a single vertex w' that is connected to all the vertices u' and v' are connected to. Vertex w' belongs to cluster $\sigma(u')$ and to all the ancestors of $\sigma(u')$ in T. The resulting clustered graph $C'(G', T')$ is easily shown to be a maximal c-planar clustered graph. In particular, the absence of separating 3-cycles in G guarantees that G' is simple and internally-triangulated. Observe that C_o and C'_o are the same graph. Hence, the inductive hypothesis applies and, for an arbitrary triangular-convex-separated drawing $\Gamma(C_o)$, there exists an internally-convex-separated drawing $\Gamma(C')$ of C' completing $\Gamma(C_o)$. Then, consider a small disk D centered at w' and consider any line l from w' to an interior point of the segment between z'_1 and z'_2. Replace w' with u' and v' so that such vertices lie on l and inside D. Connect u' and v' to their neighbors, obtaining a drawing $\Gamma(C)$ of C.

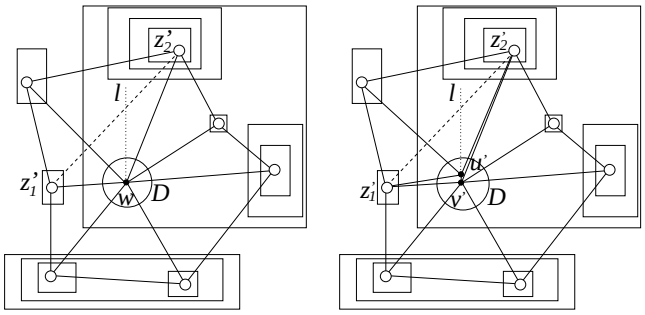


Figure 10.31: Illustration for Case 3 of the algorithm for drawing clustered graphs.

Lemma 10.19 *$\Gamma(C)$ is an internally-convex-separated drawing of C.*

Proof: In [Far48] the following property of planar straight-line drawings has been proved. In a planar straight-line drawing, for any vertex w', there exists a disk D centered at w' such that moving w' to any point inside D leaves the straight-line drawing planar. The proof of this property takes into account

the set of points from which all the neighbors of w' are *visible*, i.e., straight-line segments can be drawn without causing crossings. This property of planar straight-line drawings has been exploited in [Far48, DT88, DFP08] to argue that, in a graph G with no separating 3-cycle, an edge (u', v') that has been contracted to a single vertex w' (obtaining a graph G') can suitably replace w' so that the resulting straight-line drawing of G is planar. Here the continuity arguments used in [Far48] to prove the existence of D are still valid; however, the visibility between any point p of D and any neighbor z' of w' means that it is possible to draw a straight-line segment from p to z' not crossing any edge of G and not crossing twice the boundary of the same cluster; further, D has to be so small that it does not intersect the boundary of any cluster.

Then, the placement of u' and v' guarantees that $\Gamma(C)$ has no edge crossing and no edge-region crossing. Further, $\Gamma(C)$ has no region-region crossing, by induction. $\Gamma(C)$ is a straight-line rectangular drawing, by construction. Finally, for each internal face f of G not incident to u' and v' regions $R(\mu, \nu)$ can be drawn as in $\Gamma(C')$, since f has the same drawing in $\Gamma(C)$ and in $\Gamma(C')$; for each internal face f of G incident to u' and not to v', or viceversa, regions $R(\mu, \nu)$ can be drawn similarly to $\Gamma(C')$, since the drawings of f in $\Gamma(C)$ and in $\Gamma(C')$ differ for an arbitrary small displacement of an incident vertex (in C', face f is incident to vertex w', that replaces the one out of u' and v' that is incident to f in C); faces (u', v', z'_1) and (u', v', z'_2) are so thin that no vertex of any rectangle representing a cluster lies inside such faces, hence regions $R(\mu, \nu)$ can easily be drawn. □

It remains to prove that the case in which C is an outerclustered graph is the base case.

Lemma 10.20 *Suppose that none of Cases 1, 2, and 3 applies. Then C is an outerclustered graph.*

Proof: Suppose, for a contradiction, that none of Cases 1, 2, and 3 applies, and that C is not an outerclustered graph. By the maximality of G, $o(G)$ is delimited by a 3-cycle. By the c-planarity of C, each cluster that contains some but not all the vertices incident to $o(G)$ intersects $o(G)$ exactly twice, thus proving Property O2 of Definition 10.1.

Suppose that C contains a cluster μ not containing any vertex incident to $o(G)$. Then, C contains a minimal cluster μ' not containing any vertex incident to $o(G)$, namely $\mu' = \mu$ if μ is minimal, and μ' is any minimal cluster descendant of μ, if μ is not minimal. If μ' contains exactly one vertex v, then μ' is a minimal cluster containing only v, and Case 1 applies. If μ' contains more

Algorithm 16 DRAWING CLUSTERED GRAPHS

Require: $C(G,T)$ and $\Gamma(C_o)$.
Ensure: An internally-convex-separated drawing $\Gamma(C)$ of C completing $\Gamma(C_o)$.

1: **if** a minimal cluster μ exists containing exactly one internal vertex v and no external vertex **then**
2: $C' \leftarrow$ the graph obtained from C by removing μ;
3: $\Gamma(C') \leftarrow$ the drawing obtained by applying Algorithm 16 on C' and $\Gamma(C_o)$;
4: $\Gamma(C) \leftarrow$ the drawing obtained by inserting μ in $\Gamma(C')$ as a small rectangle containing v;
5: **else**
6: **if** there exists a separating 3-cycle $f = (u', v', z')$ **then**
7: $G^1 \leftarrow$ the subgraph of G induced by the vertices external to or belonging to (u', v', z');
8: $T^1 \leftarrow$ the subtree of T whose clusters contain vertices of G^1;
9: $C^1 \leftarrow (G^1, T^1)$;
10: $G^2 \leftarrow$ the subgraph of G induced by the vertices internal to or belonging to (u', v', z');
11: $T^2 \leftarrow$ the subtree of T whose clusters contain vertices of G^2;
12: $C^2 \leftarrow (G^2, T^2)$;
13: $\Gamma(C^1) \leftarrow$ the drawing obtained by applying Algorithm 16 on C^1 and $\Gamma(C_o)$;
14: $\Gamma(C_f) \leftarrow$ the drawing of C_f in $\Gamma(C^1)$;
15: $\Gamma(C^2) \leftarrow$ the drawing obtained by applying Algorithm 16 on C^2 and $\Gamma(C_f)$;
16: $\Gamma(C) \leftarrow$ the drawing obtained by plugging $\Gamma(C^2)$ into $\Gamma(C^1)$;
17: **else**
18: **if** an edge (u', v') exists not incident to $o(G)$ such that $\sigma(u') = \sigma(v')$ **then**
19: z'_1 and $z'_2 \leftarrow$ the common neighbors of u' and v';
20: $G' \leftarrow$ the graph obtained from G by contracting (u', v') to a vertex w';
21: $T' \leftarrow$ the tree obtained from T by assigning w' to $\sigma(u')$;
22: $C' \leftarrow (G', T')$;
23: $\Gamma(C') \leftarrow$ the drawing obtained by applying Algorithm 16 on C' and $\Gamma(C_o)$;
24: $l \leftarrow$ a line from w' to an interior point of $\overline{z'_1 z'_2}$ in $\Gamma(C')$;
25: $\Gamma(C) \leftarrow$ the drawing obtained from $\Gamma(C')$ by replacing w' with u' and v' so that u' and v' lie on l inside a small disk centered at w';
26: **else**
27: $\Gamma(C) \leftarrow$ the drawing obtained by applying Algorithm 12 on C and $\Gamma(C_o)$;
28: **end if**
29: **end if**
30: **end if**
31: **return** $\Gamma(C)$;

than one vertex, then, by the c-connectivity of C, there exists at least one edge (u, v) such that $\sigma(u) = \sigma(v) = \mu'$. If (u, v) is an edge of a separating 3-cycle, then Case 2 applies. Otherwise, Case 3 applies. This proves Property O1.

Suppose that C contains an edge (u, v) such that $\sigma(u) = \sigma(v)$ and suppose that at least one out of u and v is an internal vertex of G. If edge (u, v) belongs to a separating 3-cycle, then Case 2 applies. Otherwise, Case 3 applies. This proves Property O3. □

Hence, we get the following:

Theorem 10.3 *Let $C(G, T)$ be a maximal c-planar clustered graph. Then, for every triangular-convex-separated drawing $\Gamma(C_o)$ of C_o, there exists an internally-convex-separated drawing $\Gamma(C)$ of C completing $\Gamma(C_o)$.*

A pseudo-code description of the algorithm for drawing a clustered graph C is presented in Algorithm 16.

10.6 Conclusions

In this chapter we have shown that every c-planar clustered graph admits a c-planar straight-line rectangular drawing. Actually, the algorithms we proposed do not exploit at all the fact that clusters are drawn as rectangles. The only property that must be satisfied by each region representing a cluster for the algorithm to work is that any edge between a vertex inside the cluster and a vertex outside the cluster should cross its boundary exactly once. Hence, the algorithm we proposed can be modified in order to construct a c-planar straight-line drawing of a given clustered graph for an arbitrary assignment of convex shapes to the clusters (actually, *star-shaped polygons* are more generally feasible, i.e., polygons that have a set of points, called *kernel*, from which it is possible to draw edges towards all the vertices of the polygon without crossing its sides).

The algorithm we described uses real coordinates, hence it requires exponential area to be implemented in a system with a finite resolution rule. However, this drawback is unavoidable, since it has been proved by Feng *et al.* [FCE95a] that there exist clustered graphs requiring exponential area in any straight-line drawing in which clusters are represented by convex regions. We believe worth of interest studying the same problem for flat clustered graphs.

Open Problem 10.1 *What are the area requirements for convex (rectangular) clustered drawings of flat clustered graphs?*

Part VI

Publications and Bibliography

Chapter 11

Other Research Activities

Simultaneously with the research for the development of this thesis, other topics in the area of Graph Drawing have been dealt with:

- *Right-Angle Crossing Drawings.* Right Angle Crossing (RAC) drawings are polyline drawings where each crossing forms four right angles. RAC drawings have been introduced in [DEL09] because, even if experimental results show that the human performance in path tracing tasks is negatively correlated to the number of edge crossings [Pur00, PCA02, WPCM02], further cognitive experiments in graph visualization provided evidence that the number of crossings does not decrease the readability of the drawing if the edges cross at right angles [Hua08, HHE08].

  As the class of graphs that can be drawn with right-angle crossings includes the class of planar graphs, in [ACD+09] we investigate to what extent RAC drawings can help in overcoming the limitations of widely adopted planar graph drawing conventions, providing both positive and negative results. First, we prove that there exist acyclic planar digraphs not admitting any straight-line upward RAC drawing and that the corresponding decision problem is NP-hard. Also, we show digraphs whose straight-line upward RAC drawings require exponential area. Second, we study if RAC drawings allow to draw bounded-degree graphs with lower curve complexity than the one required by more constrained drawing conventions. We prove that every graph with vertex-degree at most 6 (at most 3) admits a RAC drawing with curve complexity 2 (resp. 1) and with quadratic area. Third, we consider a natural non-planar gen-

eralization of planar embedded graphs. Here we give bounds for curve complexity and area different from the ones known for planar embeddings.

- *Acyclic 3-Coloring of Planar Graphs.* A *coloring* of a graph is an assignment of *colors* to the vertices such that no two adjacent vertices have the same color. A *k-coloring* is a coloring using k colors. Planar graph colorings have been widely studied from both a combinatorial and an algorithmic point of view. The existence of a 4-coloring for every planar graph, proved by Appel and Haken [AH77, AHK77], is one of the most famous results in Graph Theory.

  An *acyclic coloring* is a coloring with *no bichromatic cycle*. An *acyclic k-coloring* is an acyclic coloring using k colors. Acyclic colorings have been deeply investigated in the literature. From an algorithmic point of view, Kostochka proved in [Kos78] that deciding whether a graph admits an acyclic 3-coloring is NP-hard. From a combinatorial point of view, the most interesting result is perhaps the one proved by Alon *et al.* in [AMR91], namely that every graph with degree Δ can be acyclically colored with $O(\Delta^{4/3})$ colors, while there exist graphs requiring $\Omega(\Delta^{4/3}/\sqrt[3]{\log \Delta})$ colors in any acyclic coloring. Acyclic colorings of planar graphs have been first considered in 1973 by Grünbaum, who proved in [Gru73] that there exist planar graphs requiring 5 colors in any acyclic coloring. The same lower bound holds even for 3-degenerate bipartite planar graphs [KM76]. Such a bound is tight, as proved by Borodin [Bor79].

  In [AF10], we study the planar graphs that admit an acyclic 3-coloring. We show that testing acyclic 3-colorability is NP-hard, even for planar graphs of maximum degree 4, and we show that there exist infinite classes of cubic planar graphs that are not acyclically 3-colorable. Further, we show that every planar graph has a subdivision with one vertex per edge that admits an acyclic 3-coloring. Finally, we show that every series-parallel graph admits an acyclic 3-coloring and we give a linear-time algorithm for recognizing whether every 3-coloring of a series-parallel graph is acyclic.

Publications

Journal Publications

1. P.Angelini, G. Di Battista, M. Patrignani. Finding a Minimum-Depth Embedding of a Planar Graph in $O(n^4)$ Time. *Algorithmica*, 2010. In print.

2. P.Angelini, F.Frati, L.Grilli. An Algorithm to Construct Greedy Drawings of Triangulations. *Special Issue of Journal of Graph Algorithms and Applications, vol.14(1), pages 19-51*, 2010.

Conference Publications

1. P. Angelini, F. Frati. Acyclically 3-Colorable Planar Graphs. *In Workshop on Algorithms and Computation (WALCOM '10).* To appear, 2010.

2. P. Angelini, G. Di Battista, F. Frati, V. Jelínek, J. Kratochvíl, M. Patrignani, I. Rutter. Testing Planarity of Partially Embedded Graphs. *In Symposium On Discrete Algorithms (SODA '10), ACM-SIAM.* To appear, 2010.

3. P. Angelini, G. Di Battista, F. Frati. Succinct Greedy Drawings Do Not Always Exist. *In 17th International Symposium on Graph Drawing (GD '09),* volume 5849 of LNCS, pages 171-182, 2009.

4. P. Angelini, L. Cittadini, G. Di Battista, W. Didimo, F. Frati, M. Kaufmann, A. Symvonis. On the Perspectives Opened by Right Angle Crossing Drawings. *In 17th International Symposium on Graph Drawing (GD '09),* volume 5849 of LNCS, pages 21-32, 2009.

5. P. Angelini, F. Frati, M. Patrignani. Splitting Clusters To Get C-Planarity. *In 17th International Symposium on Graph Drawing (GD '09),* volume 5849 of LNCS, pages 57-68, 2009.

6. P. Angelini, F. Frati, M. Kaufmann. Straight-line Rectangular Drawings of Clustered Graphs. *In 11th Algorithms and Data Structures Symposium (WADS '09),* volume 5664 of LNCS, pages 25-36, 2009.

7. P. Angelini, F. Frati, L. Grilli. An Algorithm to Construct Greedy Drawings of Triangulations. *In 16th International Symposium on Graph Drawing (GD '08),* volume 5417 of LNCS, pages 26-37, 2008.

8. P.Angelini, P.F. Cortese, G. Di Battista, M. Patrignani. Topological Morphing of Planar Graphs. *In 16th International Symposium on Graph Drawing (GD '08),* volume 5417 of LNCS, pages 145-156, 2008.

9. P. Angelini, G. Di Battista, M. Patrignani. Computing a Minimum-Depth Planar Graph Embedding in $O(n^4)$ Time. *In 10th Algorithms and Data Structures Symposium (WADS '07),* volume 4619 of LNCS, pages 287-299, 2007.

Technical Reports

1. P. Angelini, G. Di Battista, M. Patrignani. Computing a Minimum-Depth Planar Graph Embedding in $O(n^4)$ Time. Technical Report RT-DIA-116-2007, Dept. of Computer Science and Automation, University of Roma Tre, 2007.
http://dipartimento.dia.uniroma3.it/ricerca/rapporti/rt/2007-116.pdf

2. P. Angelini, P. F. Cortese, G. Di Battista, M. Patrignani. Topological Morphing of Planar Graphs. Technical Report RT-DIA-134-2008, Dept. of Computer Science and Automation, Roma Tre University, 2008.
http://dipartimento.dia.uniroma3.it/ricerca/rapporti/rt/2008-134.pdf

3. P. Angelini, F. Frati, L. Grilli. An Algorithm to Construct Greedy Drawings of Triangulations. Technical Report RT-DIA-140-2009, Dept. of Computer Science and Automation, Roma Tre University, 2009.
http://dipartimento.dia.uniroma3.it/ricerca/rapporti/rt/2009-140.pdf

4. P. Angelini, F. Frati, M. Kaufmann. Straight-line Rectangular Drawings of Clustered Graphs. Technical Report RT-DIA-144-2009, Dept. of Computer Science and Automation, Roma Tre University, 2009.
http://dipartimento.dia.uniroma3.it/ricerca/rapporti/rt/2009-144.pdf

5. P. Angelini, F. Frati. Acyclically 3-Colorable Planar Graphs. Technical Report RT-DIA-147-2009, Dept. of Computer Science and Automation, University of Roma Tre, 2009.
http://dipartimento.dia.uniroma3.it/ricerca/rapporti/rt/2009-147.pdf

6. P. Angelini, G. Di Battista, F. Frati. Succinct Greedy Drawings May Be Unfeasible. Technical Report RT-DIA-148-2009, Dept. of Computer Science and Automation, University of Roma Tre, 2009.
http://dipartimento.dia.uniroma3.it/ricerca/rapporti/rt/2009-148.pdf

7. P. Angelini, M. Geyer, M. Kaufmann, D. Neuwirth. On a Tree and a Path with no Geometric Simultaneous Embedding. CoRR abs/1001.0555v1, 2010. http://arxiv.org/abs/1001.0555v1

Bibliography

[AA04] A. Auyeung and A. Abraham. Estimating genome reversal distance by genetic algorithm. *CoRR*, cs.AI/0405014, 2004.

[ACBP08] P. Angelini, P. F. Cortese, G. Di Battista, and M. Patrignani. Topological morphing of planar graphs. In I. G. Tollis and M. Patrignani, editors, *International Symposium on Graph Drawing (GD '08)*, volume 5417 of *LNCS*, pages 145–156, 2008.

[ACD+09] P. Angelini, L. Cittadini, G. Di Battista, W. Didimo, F. Frati, M. Kaufmann, and A. Symvonis. On the perspectives opened by right angle crossing drawings. In D. Eppstein and E. R. Gansner, editors, *International Symposium on Graph Drawing (GD '09)*, volume 5849 of *LNCS*, pages 21–32, 2009.

[ADF09] P. Angelini, G. Di Battista, and F. Frati. Succinct greedy drawings do not always exist. In D. Eppstein and E. R. Gansner, editors, *International Symposium on Graph Drawing (GD '09)*, volume 5849 of *LNCS*, pages 171–182, 2009.

[ADF+10] P. Angelini, G. Di Battista, F. Frati, V. Jelinek, J. Kratochvil, M. Patrignani, and I. Rutter. Testing planarity of partially embedded graphs. In *Symposium On Discrete Algorithms (SODA '10)*, 2010. To appear.

[ADP07] P. Angelini, G. Di Battista, and M. Patrignani. Computing a minimum-depth planar graph embedding in $o(n^4)$ time. In F. Dehne, J.-R. Sack, and N. Zeh, editors, *Workshop on Algorithms and Data Structures (WADS '07)*, volume 4619 of *LNCS*, pages 287–299, 2007.

[ADP10] P. Angelini, G. Di Battista, and M. Patrignani. Finding a minimum-depth planar graph embedding in $o(n^4)$ time. *Algorithmica*, 2010. In print.

[AF10] P. Angelini and F. Frati. Acyclically 3-colorable planar graphs. In *Workshop on Algorithms and Computation (WALCOM '10)*, LNCS, 2010. To appear.

[AFG08] P. Angelini, F. Frati, and L. Grilli. An algorithm to construct greedy drawings of triangulations. In M. Patrignani and I. Tollis, editors, *Graph Drawing (GD '08)*, volume 5417 of *LNCS*, pages 26–37, 2008.

[AFG10] P. Angelini, F. Frati, and L. Grilli. An algorithm to construct greedy drawings of triangulations. *Journal of Graph Algorithms and Applications*, 14(1):19–51, 2010.

[AFK09] P. Angelini, F. Frati, and M. Kaufmann. Straight-line rectangular drawings of clustered graphs. In F. K. H. A. Dehne, M. L. Gavrilova, J.-R. Sack, and C. Toth, editors, *11th Algorithms and Data Structures Symposium (WADS '09)*, volume 5664 of *LNCS*, pages 25–36, 2009.

[AFP09] P. Angelini, F. Frati, and M. Patrignani. Splitting clusters to get c-planarity. In *International Symposium on Graph Drawing (GD '09)*, volume 5849 of *LNCS*, pages 57–68, 2009.

[AGKN10] P. Angelini, M. Geyer, M. Kaufmann, and D. Neuwirth. On a tree and a path with no geometric simultaneous embedding. *CoRR*, abs/1001.0555v1, 2010.

[AH77] K. Appel and W. Haken. Every planar map is 4-colorable. i. discharging. *Illinois J. Math.*, 21(3):429–490, 1977.

[AHK77] K. Appel, W. Haken, and J. Koch. Every planar map is 4-colorable. ii. reducibility. *Illinois J. Math.*, 21(3):491–567, 1977.

[AHU83] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, 1983.

[AMR91] N. Alon, C. McDiarmid, and B. A. Reed. Acyclic coloring of graphs. *Random Struct. Algorithms*, 2(3):277–288, 1991.

[Bak94] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41:153–180, 1994.

[BBF04] C. Bachmaier, F. J. Brandenburg, and M. Forster. Track planarity testing and embedding. In P. Van Emde Boas, J. Pokorný, M. Bieliková, and J. Štuller, editors, *Proc. SOFSEM 2004*, volume 2, pages 3–17. MatFyzPress, 2004.

[BCD+07] P. Brass, E. Cenek, C. Duncan, A. Efrat, C. Erten, D. Ismailescu, S. Kobourov, A. Lubiw, and J. Mitchell. On simultaneous planar graph embeddings. *Computationl Geometry*, 36(2):117–130, 2007.

[BCGG06] M. Ben-Chen, C. Gotsman, and S. J. Gortler. Routing with guaranteed delivery on virtual coordinates. In *Canadian Conference on Computational Geometry (CCCG '06)*, 2006.

[BCGW07] M. Ben-Chen, C. Gotsman, and C. Wormser. Distributed computation of virtual coordinates. In J. Erickson, editor, *Symposium on Computational Geometry (SoCG '07)*, pages 210–219, 2007.

[BDD00] P. Bertolazzi, G. Di Battista, and W. Didimo. Computing orthogonal drawings with the minimum number of bends. *IEEE Transactions on Computers*, 49:8:826–840, 2000.

[BEF+07] U. Brandes, C. Erten, J. Fowler, F. Frati, M. Geyer, C. Gutwenger, S.-H. Hong, M. Kaufmann, S. Kobourov, G. Liotta, P. Mutzel, and A. Symvonis. Colored simultaneous geometric embeddings. In G. Lin, editor, *International Computing and Combinatorics Conference (COCOON)*, volume 4598 of *LNCS*, pages 254–263, 2007.

[BF07] G. Di Battista and F. Frati. Efficient c-planarity testing for embedded flat clustered graphs with small faces. In S. H. Hong, T. Nishizeki, and W. Quan, editors, *Graph Drawing (GD '07)*, volume 4875 of *LNCS*, pages 291–302, 2007.

[BFM07] N. Bonichon, S. Felsner, and M. Mosbah. Convex drawings of 3-connected plane graphs. *Algorithmica*, 47(4):399–420, 2007.

[BL76] K. Booth and G. Lueker. Testing for the consecutive ones property interval graphs and graph planarity using PQ-tree algorithms. *J. Comp. Syst. Sci.*, 13:335–379, 1976.

[BLS05] T. C. Biedl, A. Lubiw, and M. J. Spriggs. Morphing planar graphs while preserving edge directions. In P. Healy and N. S. Nikolov, editors, *Graph Drawing, (GD'05)*, volume 3843 of *LNCS*, pages 13–24. Springer, 2005.

[BM76] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Macmillan, London, United Kingdom, 1976.

[BM88] D. Bienstock and C. L. Monma. On the complexity of covering vertices by faces in a planar graph. *SIAM-Journal on Computing*, 17:53–76, 1988.

[BM90] D. Bienstock and C. L. Monma. On the complexity of embedding planar graphs to minimize certain distance measures. *Algorithmica*, 5(1):93–109, 1990.

[BM04] J. M. Boyer and W. J. Myrvold. On the cutting edge: simplified $O(n)$ planarity by edge addition. *Journal of Graph Algorithms and Applications*, 8(3):241–273, 2004.

[BMY01] D. A. Bader, B. M. E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

[Bor79] O. V. Borodin. On acyclic colourings of planar graphs. *Discr. Math*, 25:211–236, 1979.

[Cai44] S. S. Cairns. Deformations of plane rectilinear complexes. *American Math. Monthly*, 51(3):247–252, 1944.

[Cap97] A. Caprara. Sorting by reversals is difficult. In *RECOMB '97: Proceedings of the first annual international conference on Computational molecular biology*, pages 75–83. ACM, 1997.

[CB05] P. F. Cortese and G. Di Battista. Clustered planarity. In *Symposium on Computational Geometry (SoCG '05)*, pages 32–34, 2005.

[CBF+08] P. F. Cortese, G. Di Battista, F. Frati, M. Patrignani, and M. Pizzonia. C-planarity of c-connected clustered graphs. *Journal of Graph Algorithms and Applications*, 12(2):225–262, 2008.

[CDPP05a] P. F. Cortese, G. Di Battista, M. Patrignani, and M. Pizzonia. Clustering cycles into cycles of clusters. *Journal of Graph Algorithms and Applications*, 9(3):391–413, 2005.

[CDPP05b] P. F. Cortese, G. Di Battista, M. Patrignani, and M. Pizzonia. On embedding a cycle in a plane graph. In P. Healy and N. S. Nikolov, editors, *Graph Drawing (GD '05)*, volume 3843 of *LNCS*, pages 49–60, 2005.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill Book Company, Boston, MA, 2001.

[CN98] M. Chrobak and S.-I. Nakano. Minimum-width grid drawings of plane graphs. *Computat. Geom. Th. Appl.*, 11:29–54, 1998.

[CvKL+09] S. Cabello, M. van Kreveld, G. Liotta, H. Meijer, B. Speckmann, and K. Verbeek. Geometric simultaneous embeddings of a graph and a matching. In D. Eppstein and E. R. Gansner, editors, *International Symposium on Graph Drawing (GD)*, volume 5849 of *LNCS*, 2009.

[CW06] S. Cornelsen and D. Wagner. Completely connected clustered graphs. *Journal of Discrete Algorithms*, 4(2):313–323, 2006.

[Dah98] E. Dahlhaus. A linear time algorithm to recognize clustered graphs and its parallelization. In C. L. Lucchesi and A. V. Moura, editors, *Latin American Theoretical Informatics (LATIN '98)*, LNCS, pages 239–248, 1998.

[DDF07] G. Di Battista, G. Drovandi, and F. Frati. How to draw a clustered tree. In F. K. H. A. Dehne, J. Sack, and N. Zeh, editors, *WADS '07*, pages 89–101, 2007.

[DDLM05] E. Di Giacomo, W. Didimo, G. Liotta, and H. Meijer. Computing radial drawings on the minimum number of circles. *Journal of Graph Algorithms and Applications*, 9:365–389, 2005.

[DDv+07] E. Di Giacomo, W. Didimo, M. van Kreveld, G. Liotta, and B. Speckmann. Matched drawings of planar graphs. In S.-H. Hong, T. Nishizeki, and W. Quan, editors, *International Symposium on Graph Drawing (GD)*, volume 4875 of *LNCS*, pages 183–194, 2007.

[DEL09] W. Didimo, P. Eades, and G. Liotta. Drawing graphs with right angle crossings. In *WADS '09*, LNCS, 2009. To appear.

[DETT99] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[dFdMR06] H. de Fraysseix, P. O. de Mendez, and P. Rosenstiehl. Trémaux trees and planarity. *Int. J. Found. Comput. Sci.*, 17:1017–1030, 2006.

[DFP08] G. Di Battista, F. Frati, and M. Patrignani. Non-convex representations of graphs. In I. Tollis and M. Patrignani, editors, *GD '08*, pages 390–395, 2008.

[Dha08] R. Dhandapani. Greedy drawings of triangulations. In S. T. Huang, editor, *Symposium on Discrete Algorithms (SODA '08)*, pages 102–111, 2008.

[Die05] R. Diestel. *Graph Theory*. Springer, Heidelberg, Germany, 2005.

[DL07] E. Di Giacomo and G. Liotta. Simultaneous embedding of outerplanar graphs, paths, and cycles. *Int. J. Comput. Geom. Appl.*, 17(2):139–160, 2007.

[DLL95] G. Di Battista, W. Lenhart, and G. Liotta. Proximity drawability: a survey. In R. Tamassia and I. G. Tollis, editors, *GD '94*, volume 894 of *LNCS*, pages 328–339, 1995.

[DLT84] D. Dolev, F. T. Leighton, and H. Trickey. Planar embedding of planar graphs. *Advances in Computing Research - Volume 2: VLSI Theory*, 1984.

[dO] H. de Fraysseix and P. Ossona de Mendez. P.I.G.A.L.E - Public Implementation of a Graph Algorithm Library and Editor. SourceForge project page `http://sourceforge.net/projects/pigale`.

[Dor02] C. Dornheim. Planar graphs with topological constraints. *J. Graph Alg. Appl.*, 6(1):27–66, 2002.

[dPP88] H. de Fraysseix, J. Pach, and R. Pollack. Small sets supporting fáry embeddings of planar graphs. In *Symposium on Theory of Computing (STOC '88)*, pages 426–433, 1988.

[dPP90] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[dR82] H. de Fraysseix and P. Rosenstiehl. A depth-first-search characterization of planarity. *Ann. Discr. Math.*, 13:75–80, 1982.

[DT88] G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61:175–198, 1988.

[DT90] G. Di Battista and R. Tamassia. On-line graph algorithms with SPQR-trees. In M. Paterson, editor, *International Colloquium on Automata, Languages and Programming (ICALP '90)*, LNCS, pages 598–611, 1990.

[DT96a] G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with spqr-trees. *Algorithmica*, 15(4):302–318, 1996.

[DT96b] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM Journal on Computing*, 25(5):956–997, 1996.

[DTT92] G. Di Battista, R. Tamassia, and I. G. Tollis. Area requirement and symmetry display of planar upward drawings. *Discrete & Computational Geometry*, 7:381–401, 1992.

[dvKOS00] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry - Algorithms and Applications*. Springer, Heidelberg, Germany, 2000.

[EBFK09] A. Estrella-Balderrama, J. Fowler, and S. Kobourov. Characterization of unlabeled level planar trees. *Computational Geometry: Theory and Applications*, 42(6-7):704–721, 2009.

[EBGJ+07] 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.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer, New York, NY, 1987.

[EFL96] P. Eades, Q. W. Feng, and X. Lin. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. In S. North, editor, *GD '96*, pages 113–128, 1996.

[EFLN06] P. Eades, Q. Feng, X. Lin, and H. Nagamochi. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. *Algorithmica*, 44(1):1–32, 2006.

[EFN99] P. Eades, Q. Feng, and H. Nagamochi. Drawing clustered graphs on an orthogonal grid. *Journal of Graph Algorithms and Applications*, 3(4):3–29, 1999.

[EG08] D. Eppstein and M. Goodrich. Succinct greedy graph drawing in the hyperbolic plane. In M. Patrignani and I. Tollis, editors, *Graph Drawing 2008*, pages 14–25, 2008.

[EK04] C. Erten and S. Kobourov. Simultaneous embedding of planar graphs with few bends. In J. Pach, editor, *International Symposium on Graph Drawing (GD)*, volume 3383 of *LNCS*, pages 195–205, 2004.

[EKP03] C. Erten, S. G. Kobourov, and C. Pitta. Intersection-free morphing of planar graphs. In *11th Symposium on Graph Drawing (GD'03)*, pages 320–331, 2003.

[ET76] S. Even and R. E. Tarjan. Computing an st-numbering. *Theor. Comp. Sci.*, 2:339–344, 1976.

[Eve79] S. Even. *Graph Algorithms*. Computer Science Press, Potomac, Maryland, 1979.

[Far48] I. Fary. On straight line represention of planar graphs. *Acta. Sci. Math.*, 11:229–233, 1948.

[FCE95a] Q. Feng, R. F. Cohen, and P. Eades. How to draw a planar clustered graph. In D. Du and M. Li, editors, *Computing and Combinatorics Conference (COCOON '95)*, volume 959 of *LNCS*, pages 21–30, 1995.

[FCE95b] Q. Feng, R. F. Cohen, and P. Eades. Planarity for clustered graphs. In P. G. Spirakis, editor, *European Symposium on Algorithms (ESA '95)*, volume 979 of *LNCS*, pages 213–226, 1995.

[Fen97] Q. Feng. *Algorithms for Drawing Clustered Graphs*. PhD thesis, The University of Newcastle, Australia, 1997.

[FG99] M. Floater and C. Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics*, 101:117–129, 1999.

[FGJ+08] 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.

[Fia03] J. Fiala. NP-completeness of the edge precoloring extension problem on bipartite graphs. *J. Graph Theory*, 43(2):156–160, 2003.

[FJKS08] 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, *International Workshop on Graph Theoretic Concepts in Computer Science (WG)*, volume 5344 of *LNCS*, pages 146–158, 2008.

[FK07a] J. Fowler and S. Kobourov. Characterization of unlabeled level planar graphs. In S.-H. Hong, T. Nishizeki, and W. Quan, editors, *International Symposium on Graph Drawing (GD)*, volume 4875 of *LNCS*, pages 37–49, 2007.

[FK07b] J. Fowler and S. Kobourov. Minimum level nonplanar patterns for trees. In S.-H. Hong, T. Nishizeki, and W. Quan, editors, *International Symposium on Graph Drawing (GD)*, volume 4875 of *LNCS*, pages 69–75, 2007.

[FKK07] F. Frati, M. Kaufmann, and S. Kobourov. Constrained simultaneous and near simultaneous embeddings. In S.-H. Hong, T. Nishizeki, and W. Quan, editors, *International Symposium on Graph Drawing (GD)*, volume 4875 of *LNCS*, pages 268–279, 2007.

[FP07] F. Frati and M. Patrignani. A note on minimum area straight-line drawings of planar graphs. In S. H. Hong and T. Nishizeki, editors, *Graph Drawing (GD '07)*, LNCS, pages 339–344, 2007.

[Fra06] F. Frati. Embedding graphs simultaneously with fixed edges. In M. Kaufmann and D. Wagner, editors, *International Symposium on Graph Drawing (GD)*, volume 4372 of *LNCS*, pages 108–113, 2006.

[GJ77] M. Garey and D. Johnson. The rectilinear steiner tree problem is NP-complete. *SIAM J. Appl. Math.*, 32(4):826–834, 1977.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GJL+02] C. Gutwenger, M. Jünger, S. Leipert, P. Mutzel, M. Percan, and R. Weiskircher. Advances in c-planarity testing of clustered graphs. In S. G. Kobourov and M. T. Goodrich, editors, *Graph Drawing (GD '02)*, volume 2528 of *LNCS*, pages 220–235, 2002.

[GJP+06] E. Gassner, M. Jünger, M. Percan, M. Schaefer, and M. Schulz. Simultaneous graph embeddings with fixed edges. In F. V. Fomin, editor, *International Workshop on Graph Theoretic Concepts in Computer Science (WG)*, volume 4271 of *LNCS*, pages 325–335, 2006.

[GKM08] 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.

[GKV09] M. Geyer, M. Kaufmann, and I. Vrt'o. Two trees which are self-intersecting when drawn simultaneously. *Discrete Mathematics*, 309(7):1909 – 1916, 2009.

[GLS05] M. T. Goodrich, G. S. Lueker, and J. Z. Sun. C-planarity of extrovert clustered graphs. In P. Healy and N. Nikolov, editors, *Graph Drawing (GD '05)*, volume 3843 of *LNCS*, pages 211–222, 2005.

[GM00] 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.

[GR94] Z. Gao and R. B. Richter. 2-walks in circuit graphs. *J. Comb. Theory, Ser. B*, 62(2):259–267, 1994.

[GRS90] R. L. Graham, B. L. Rothschild, and J. H. Spencer. *Ramsey Theory*. John Wiley & Sons, 1990.

[Gru73] B. Grunbaum. Acyclic colorings of planar graphs. *Israel J. Math.*, 14:390–408, 1973.

[GS01] C. Gotsman and V. Surazhsky. Guaranteed intersection-free polygon morphing. *Computers and Graphics*, 25:67–75, 2001.

[GS09a] S. K. Ghosh and K. Sinha. On convex greedy embedding conjecture for 3-connected planar graphs. In M. Kutylowski, W. Charatonik, and M. Gebala, editors, *Symposium on Fundamentals of Computation Theory (FCT '09)*, volume 5699 of *LNCS*, pages 145–156, 2009.

[GS09b] M. T. Goodrich and D. Strash. Succinct greedy geometric routing in the Euclidean plane. In Y. Dong, D.-Z. Du, and O. Ibarra, editors, *International Symposium on Algorithms and Computation (ISAAC '09)*, LNCS, 2009. to appear.

[GT02] M. T. Goodrich and R. Tamassia. *Algorithm Design*. John Wiley and Sons, New York, NY, 2002.

[Hal91] J. H. Halton. On the thickness of graphs of given degree. *Information Sciences*, 54(3):219–238, 1991.

[Har72] F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1972.

[HHE08] W. Huang, S.-H. Hong, and P. Eades. Effects of crossing angles. In *PacificVis*, pages 41–46, 2008.

[HN09] S.-H. Hong and H. Nagamochi. Convex drawings of hierarchical planar graphs and clustered planar graphs. *Journal of Discrete Algorithms*, 2009.

[HP66] F. Harary and G. Prins. The block-cutpoint-tree of a graph. *Publ. Math. Debr.*, 13:103–107, 1966.

[HT74] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

[HT08] B. Haeupler and R. E. Tarjan. Planarity algorithms via pq-trees (extended abstract). *Electronic Notes in Discrete Mathematics*, 31:143–149, 2008.

[Hua08] W. Huang. An eye tracking study into the effects of graph layout. *CoRR*, abs/0810.4431, 2008.

[JJKL08] V. Jelinek, E. Jelinkova, J. Kratochvil, and B. Lidicky. Clustered planarity: Embedded clustered graphs with two-component clusters. In M. Patrignani and I. Tollis, editors, *Graph Drawing (GD '08)*, volume 5417 of *LNCS*, pages 121–132, 2008.

[JKK+07] E. Jelinkova, J. Kara, J. Kratochvil, M. Pergel, O. Suchy, and T. Vyskocil. Clustered planarity: Small clusters in eulerian graphs. In S. H. Hong, T. Nishizeki, and W. Quan, editors, *Graph Drawing (GD '07)*, volume 4875 of *LNCS*, pages 303–314, 2007.

[JLP02] M. Jünger, S. Leipert, and M. Percan. Triangulating clustered graphs. Technical report, Zentrum für Angewandte Informatik Köln, Lehrstuhl Jünger, December 2002.

[JM05] M. Juvan and B. Mohar. 2-restricted extensions of partial embeddings of graphs. *European J. Comb.*, 26(3–4):339–375, 2005.

[Kam07] F. Kammer. Determining the smallest k such that g is k-outerplanar. In L. Arge, M. Hoffmann, and E. Welzl, editors, *ESA '07*, volume 4698 of *LNCS*, pages 359–370, 2007.

[Kau08] M. Kaufmann. Polynomial area bounds for MST embeddings of trees. In S. H. Hong, T. Nishizeki, and W. Quan, editors, *Graph Drawing (GD '07)*, volume 4875 of *LNCS*, pages 88–100, 2008.

[Kir88] D. G. Kirkpatrick. Establishing order in planar subdivisions. *Discr. Computat. Geom.*, 3:267–280, 1988.

[KK03] L. Kowalik and M. Kurowski. Short path queries in planar graphs in constant time. In *STOC '03*, pages 143–148, 2003.

[KKM29] B. Knaster, C. Kuratowski, and C. Mazurkiewicz. Ein beweis des fixpunktsatzes fur n dimensionale simplexe. *Fundamenta Mathematicae*, 14:132–137, 1929.

[KL08] S. G. Kobourov and M. Landis. Morphing planar graphs in spherical space. *Journal of Graph Algorithms and Applications*, 12(1):113–127, 2008.

[Kle07] R. Kleinberg. Geographic routing using hyperbolic space. In *INFOCOM '07*, pages 1902–1909. IEEE, 2007.

[KM76] A. V. Kostochka and L. S. Melnikov. To the paper of B. Grünbaum on acyclic colorings. *Discrete Math.*, 14:403–406, 1976.

[Kos78] A. V. Kostochka. *Upper Bounds of Chromatic Functions of Graphs*. PhD thesis, University of Novosibirsk (in Russian), 1978.

[KS97] J. Kratochvil and A. Sebo. Coloring precolored perfect graphs. *J. Graph Theory*, 25:207–215, 1997.

[KST97] H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. In *SODA '97*, pages 344–351. SIAM, 1997.

[Kur30] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15:271–283, 1930.

[KW01] M. Kaufmann and D. Wagner, editors. *Drawing Graphs, Methods and Models*, LNCS. Springer, 2001.

[LM08] T. Leighton and A. Moitra. Some results on greedy embeddings in metric spaces. In *Foundations of Computer Science (FOCS '08)*, pages 337–346, 2008.

[LP08] A. Lubiw and M. Petrick. Morphing planar graph drawings with bent edges. *Electronic Notes in Discrete Mathematics*, 31:45–48, 2008.

[LPS06] A. Lubiw, M. Petrick, and M. Spriggs. Morphing orthogonal planar graph drawings. In *Symposium on Discrete algorithm (SODA)*, pages 222–230. ACM, 2006.

[Lub07] A. Lubiw. Morphing planar graph drawings. In P. Bose, editor, *Canadian Conference on Computational Geometry (CCCG 2007)*, page 1, 2007.

[Moh99] B. Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discr. Math.*, 12(1):6–26, 1999.

[MS92] C. L. Monma and S. Suri. Transitions in geometric minimum spanning trees. *Discrete & Computational Geometry*, 8:265–293, 1992.

[MWD00] J. Meidanis, M.E.M.T. Walter, and Z. Dias. Reversal distance of sorting circular chromosomes. Tech. Report IC-00-23, Institute of Computing, Universidade Estadual de Campinas, 2000.

[NC88] T. Nishizeki and N. Chiba. *Planar Graphs: Theory and Algorithms*, volume 32 of *Ann. Discrete Math.* North-Holland, 1988.

[NK07] H. Nagamochi and K. Kuroya. Drawing c-planar biconnected clustered graphs. *Discrete Applied Mathematics*, 155(9):1155–1174, 2007.

[NR04] T. Nishizeki and M. S. Rahman. *Planar Graph Drawing.* Word Scientific, Singapore, 2004.

[Pat06] M. Patrignani. On extending a partial straight-line drawing. *Found. Comput. Sci.*, 17(5):1061–1069, 2006.

[PCA02] H. C. Purchase, D. A. Carrington, and J.-A. Allder. Empirical evaluation of aesthetics-based graph layout. *Empirical Software Engineering*, 7(3):233–255, 2002.

[PCJ97] H. C. Purchase, R. F. Cohen, and M. I. James. An experimental study of the basis for graph drawing algorithms. *ACM J. Exp. Alg.*, 2:4, 1997.

[Piz05] M. Pizzonia. Minimum depth graph embeddings and quality of the drawings: An experimental analysis. In P. Healy and N.S. Nikolov, editors, *Graph Drawing '05*, volume 3843 of *LNCS*, pages 397–408, 2005.

[Pou94] J. A. La Poutré. Alpha-algorithms for incremental planarity testing. In *STOC '94*, pages 706–715, 1994.

[PR05] C. H. Papadimitriou and D. Ratajczak. On a conjecture related to geometric routing. *Theoretical Computer Science*, 344(1):3–14, 2005.

[PS85] F. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer, New York, NY, 1985.

[PT00] M. Pizzonia and R. Tamassia. Minimum depth graph embedding. In M. Paterson, editor, *ESA 2000*, volume 1879 of *LNCS*, pages 356–367, 2000.

[Pur00] H. C. Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with Computers*, 13(2):147–162, 2000.

[PV04] P. Penna and P. Vocca. Proximity drawings in polynomial area and volume. *Computational Geometry*, 29(2):91–116, 2004.

[PW01] J. Pach and R. Wenger. Embedding planar graphs at fixed vertex locations. *Graphs and Combinatorics*, 17(4):717–728, 2001.

[RPSS03] A. Rao, C. H. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In D. B. Johnson, A. D. Joseph, and N. H. Vaidya, editors, *Conference on Mobile Computing and Networking (MOBICOM 2003)*, pages 96–108. ACM, 2003.

[RS84] N. Robertson and P. D. Seymour. Graph minors. III. Planar tree-width. *Journal on Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[Sch90] W. Schnyder. Embedding planar graphs on the grid. In *Symposium on Discrete Algorithms (SODA '90)*, pages 138–148. SIAM, 1990.

[SJTV08] O. Suchy, V. Jelinek, M. Tesar, and T. Vyskocil. Clustered planarity: Clusters with few outgoing edges. In M. Patrignani and I. Tollis, editors, *Graph Drawing (GD '08)*, volume 5417 of *LNCS*, pages 102–113, 2008.

[SSL03] A. Solomon, P. Sutcliffe, and R. Lister. Sorting circular permutations by reversal. In *WADS'03*, pages 319–328, 2003.

[Ste51] S. K. Stein. Convex maps. *Amer. Math. Soc.*, 2:464–466, 1951.

[Tam96] R. Tamassia. On-line planar graph embedding. *J. Algorithms*, 21(2):201–239, 1996.

[Tam98] R. Tamassia. Constraints in graph drawing algorithms. *Constraints*, 3(1):87–120, 1998.

[TBS07] E. Tannier, A. Bergeron, and M.-F. Sagot. Advances on sorting by reversals. *Discrete Appl. Math.*, 155(6-7):881–888, 2007.

[TDB88] R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst., Man and Cyber.*, 18(1):61–79, 1988.

[Tho83] C. Thomassen. Deformations of plane graphs. *Journal of Combinatorial Theory, Series B*, 34(3):244–257, 1983.

[Tho99] M. Thorup. Undirected single-source shortest path with positive integer weights in linear time. *Journal of the ACM*, 46(3):362–394, 1999.

[Tut56] W. T. Tutte. A theorem on planar graphs. *Trans. Amer. Math. Soc.*, 82(1):99–116, 1956.

[Val81] L. G. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. Comp.*, 30(2):135–140, 1981.

[Wag36] K. Wagner. Bemerkungen zum vierfarbenproblem. *Jahresbericht. German. Math.-Verein*, 2:26–32, 1936.

[Wag37] K. Wagner. Uber eine eigenschaft der ebenen komplexe. *Math. Ann.*, 114:570–590, 1937.

[Wes92] J. Westbrook. Fast incremental planarity testing. In W. Kuich, editor, *ICALP '92*, volume 623 of *LNCS*, pages 342–353, 1992.

[WPCM02] C. Ware, H. C. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002.

[ZH03] H. Zhang and X. He. Compact visibility representation and straight-line grid embedding of plane graphs. In F. K. H. A. Dehne, J. R. Sack, and M. H. M. Smid, editors, *Algorithms and Data Structures (WADS '03)*, volume 2748 of *LNCS*, pages 493–504, 2003.