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

Morphing and Visiting Drawings of Graphs

Vincenzo Roselli
Cycle XXVI

Candidate: Vincenzo Roselli	
Advisor: Prof. Giuseppe Di Battista	
Advisor: Prof. Maurizio Patrignani	
Coordinator: Prof. Stefano Panzieri	

Morphing and Visiting Drawings of Graphs

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

Roma Tre University
Department of Engineering

June 2014

COMMITTEE:
Prof. Giuseppe Di Battista
Prof. Maurizio Patrignani

REVIEWERS:
Prof. Sabine Cornelsen
Prof. Alexander Wolff

La vecchia non voleva morire perché diceva che voleva ancora imparare.

The old lady didn’t want to die because she still had much to learn, she said.

Italian proverb

Alla mia famiglia

Acknowledgments

My best acknowledgments go to my advisors: Giuseppe "Pino" Di Battista and Maurizio "Titto" Patrignani. During their lessons on Theoretical Computer Science, they had in their eyes that light -which only someone really in love with research can have- that led me towards the direction I'm following now. They taught me all I know about graphs and algorithms. Their passion, the fun they have at doing research, and the challenging problems they proposed to me have been a constant motivation to tackle harder and harder problems each time: during these three years I've never had the feeling of "being at work". It's been like playing a very exciting game, each single day.

A special thank goes to Patrizio "Muffa" Angelini: in these years he's been a third advisor, a research mate, and, above all, become a true friend.

I would like to thank Michael Kaufmann and Anna Lubiw for allowing me to collaborate with them during the very pleasant and interesting visit periods I spent at the University of Tübingen and at the University of Waterloo, respectively.

I would like to thank Sabine Cornelsen and Alexander Wolff for carefully reviewing this thesis in the very short time they had.

I would like to thank all the other people I collaborated with for the great time I had while working with them, they played a key role in my enjoying research so much and gave me the opportunity of learning a lot: Soroush Alamdari, Patrizio Angelini, Fidel Barrera-Cruz, Michael Bekos, Timothy Chan, Giordano Da Lozzo, Giuseppe Di Battista, Walter Didimo, Fabrizio Frati, Michael Kaufmann, Stephen Kobourov, Robert Krug, Anna Lubiw, Tamara Mchedlidze, Stefan Naher, Maurizio Patrignani, Sahil Singla, Claudio Squarcella, Antonios Symvonis, Bryan Wilkinson, and Stephen Wismath.

I would like to thank all the members of our research group for creating that funny atmosphere mixing friendship and collaboration that we have in our office: Massimo Candela, Marco "Church" Chiesa, Luca "Roscio" Cittadini, Giordano Da Lozzo, Marco "DB" Di Bartolomeo, Valentino Di Donato, Fabrizio "Brillo" Frati, Gabriele Lospoto, Alessandro "Mancio" Mancini, Alessandro Manfredi, Bernardo Palazzi, Maurizio Pizzonia, Massimo "Max" Rimondini, Giorgio Sadolfo, and Stefano Vissicchio.

The final, wholehearted, thank goes to my parents, my sister, the remainder of my family, and my friends: They are my life. Without them, all I did, and all I will do would be pointless to me.

Contents

Introduction

Among the most widely used data structures to represent pairwise relationships between entities, graphs play a key role. Graph applications can, indeed, be found in every field, ranging from maps to circuits, and from networks to interpersonal relationships. Visualizing a graph is probably one of the most expressive ways to describe the information encoded in it. Such an issue is addressed in the research field of *Graph Drawing*, which inherits techniques from the areas of *Graph Theory*, *Graph Algorithms*, and *Computational Geometry*. Namely, in a *drawing* of a graph each entity -called *vertex*- is usually represented by a point in the plane and each relationship -called *edge*- between two entities as a curve connecting the corresponding points. Clearly, not every drawing can be considered a good representation of the graph. Vertices and edges should be drawn in such a way that the human eye is facilitated in identifying the relationships among the entities at a glance. Namely, the drawing should be *readable*. During the years, some topological and geometric features that a drawing should satisfy in order to be easily readable have been recognized and formally characterized. The main goals of Graph Drawings are, then, creating algorithms that automatically produce drawings respecting such criteria and, possibly, defining new ones.

Planarity, that is the absence of partial or total overlapping among vertices and edges, is probably the most natural and desirable characteristic a drawing can have, as it allows a viewer to easily distinguish the curve used to represent any edge-relationship and hence to immediately recognize which entities-vertices participate in that relationship. Unfortunately, due to their topological structure, not all the graphs admit a planar drawing. In such cases, a natural requirement for the drawing is that of containing as few crossings as possible. Moreover, especially if the input graph -and hence the area of the drawing- is large, it is convenient that the points where two edges cross are easily distinguishable from the points where vertices are placed. Observe that, while planarity is a *property* that a drawing may fulfill or not, the area and the number of crossings are two examples of *measure of quality* that can be used to compare two drawings of the same graph.

From the geometric point of view, it would be preferable that edges are drawn as *straight-lines*. Edges that bend and repeatedly or abruptly change direction might sensibly decrease the readability of the drawing. In straight-line drawings, however, the information of interest for a user might not be sufficiently emphasized. In that case, edges can be represented as *poly-lines* bending only a limited number of times or having a limited number of slopes, so that the negative impact on the readability of the drawing is limited.

Other required features for a drawing of a graph to describe some meta-information, we recall the representation of groups (called *clusters*) of vertices that, aside from the relationship described by the edges, represent objects that share some properties, as in the case of *clustered graphs*. Also, in some contexts it might be required to emphasize the chains of relationships that indirectly involve pairs of entities. In this case, the placement of the vertices and the fashion in which edges are drawn should be able to "lead" the eye of a viewer from an object to another through the chain of relationships that are of interest.

Of course, some of such desired features can be in contrast with each other and hence cannot be simultaneously satisfied by a single drawing. It is easy to imagine contexts in which several drawings of the same graph, which can be substantially different from each other, are of interest to the same observer. Due to the great difference between two drawings, a considerable effort might be required to the user, while switching from one drawing to another, in "updating" the mental map he or she has of the graph. In order to support the user in this operation, a smooth transformation of a drawing into another might be desirable. Clearly, since the purpose of this transformation is to support the user in changing the focus from a drawing to another, it should introduce as few distracting elements, e.g. crossings, bends in the edges, or non-linear trajectories, as possible.

In this thesis we mainly deal with algorithms that compute *planar morphs*, i.e., transformations, of planar drawings of the same graph in which planarity is preserved at any time. We also consider the problem of constructing drawings that, at the same time, emphasize certain properties of a given graph and require small area. We mainly deal with planar *embedded* graphs, namely, graphs in which the circular order of the edges around each vertex is fixed and such that there exists a planar drawing of the graph in which such an order is maintained. We first recall some preliminary definitions on graphs and on the main data structures used for their decomposition. In Part I we present an algorithm for the construction of planar linear morphs of series-parallel graphs with a number of moves that is linear in the size of the graph, prove that such an algorithm is asymptotically optimal by providing a lower bound on the number of linear moves that are required to transform a planar drawing of a plane graph, and give an algorithm for constructing planar linear morphs of general plane graphs with a number of moves that is quadratic in the size of the graph. In Part II we consider the problem of computing drawings of graphs in which some features, namely chains of relationships between pairs of vertices and the difference between vertices and crossings in drawings of non-planar graphs, are emphasized, thus allowing the user to easily "visit" the underlying graph. Finally, in Appendix A we describe some further results on drawings of planar graphs.

In Background & Basics a preliminary definitions on graphs, their drawings, and their decomposition is given. Namely, in Chapter 1 we provide some preliminary definitions on graphs, their drawings, and the most commonly accepted aesthetic criteria, while Chapter 2 presents some of the data structures that are commonly used for decomposing planar graphs.

Part I is devoted to study the problem of computing a planar morph between pairs of planar straight-line drawings of planar embedded graphs. In Chapter 3 we give a survey of the main results on this topic that are known in the literature. In the same chapter, we give a complete description of an algorithm by Cairns that is considered a cornerstone in computing a planar morph of a graph. In Chapter 4 we provide some topological and geometric tools that will be useful in the remainder of the thesis.

In Chapter 5 we describe an algorithm for computing planar linear morphs of drawings of n-vertex series-parallel graphs in $O(n)$ steps. In Chapter 6 we prove that a linear number of moves is sometimes necessary, thus implying that the algorithm provided in the previous chapter is asymptotically optimal. In Chapter 7 we address the problem of constructing a planar morph with a polynomial number of steps for drawings of planar embedded graphs, the general setting for the problem. Chapter 8 concludes this part discussing some open problems on this topic.

In Part II we consider the problem of "visiting" a drawing of a graph. Namely, a user might want to "navigate" a drawing of a graph, that is moving the focus from a vertex to another following edges, and hence some features of the graph should be stressed in the drawing. In Chapter 9 we deal with the problem of constructing *monotone* drawings of plane graphs. A drawing of a graph is monotone if, for every pair of vertices, there exists a path whose vertices are placed in such a way that, walking through this path from an endpoint to another, the user gets closer (according to some measures) to the destination at each step. The study of this problem is well motivated by human subject experiments by Huang *et al.* [HEH09], who showed that the "geodesic path tendency" (paths following a given direction) is important in comprehending the underlying graph. We show that every planar embedded graph admits a monotone drawing in which each edge is represented by at most three straight-line segments and prove that such a number of bends is sometimes necessary. We also prove that outerplane and biconnected planar embedded graphs admit straight-line monotone drawings. Chapter 10 deals with non-planar graphs. Namely, we propose a new model in which edges are drawn as poly-lines composed of vertical, horizontal, and diagonal segments. The aim of such a model is to emphasize the difference between crossings and vertices in drawings of large graphs by modeling crossings as intersections of diagonal segments. We prove that every graph with max-degree 4 admits a drawing respecting this model and provide an algorithm to construct such drawings in polynomial area. We also prove a lower bound on the minimum number of bends in any drawing of a graph in which the circular order of the edges around each vertex is fixed, and show that a drawing in which the number of bends is minimized might require an area that is exponential in the number of the vertices of the graph. Chapter 11 concludes this part discussing some open problems on these topics.

In Appendix A we study the problem of drawing a graph on a given set of points and define the structure of a point set onto which every *simply-nested* planar graph can be drawn. We also study the problem of drawings *clustered graphs* and, by relaxing some constraints, we give the first non-trivial necessary condition for a clustered graph to be *clustered planar*.

Background & Basics

Chapter 1

Preliminaries

In this chapter, we give some preliminary definitions about graphs 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., [Har69, BM76, CN88, Die05]), to books on Algorithms (e.g., [Eve79, AHU83, CLRS09, GT09]), and to books on Computational Geometry (e.g., [PS85, Ede87, dCvO08]). The book of Di Battista, Eades, Tamassia, and Tollis [DETT99] is usually considered as *the book* on Graph Drawing. Other excellent books that specifically deal with Graph Drawing are [KW01, NR04]. The chapter is structured as follows. In Section 1.1 we give some preliminary definitions on graphs in general. In Section 1.2 we focus on planar graphs and, in Section 1.3, characterize some notable classes of planar graphs we deal with in the remainder of the thesis. Finally, in Section 1.4 we recall some drawing conventions and æsthetic criteria.

1.1 Basic Definitions

A *graph* G is a pair (V, E), where V is a set of elements called *vertices*, and E is a multiset of *unordered* pairs of vertices, called *edges*. The vertices v and w composing a pair $e = (v, w) \in E$ are *incident* to e, and edge e is *incident* to v and w. 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. The *end-vertices* of an edge (v, w) are vertices v and w, which are also said to be *neighbors*. The *degree of a vertex* v is the number of its incident edges (or, equivalently, the number of its neighbors) and is denoted by $\deg(v)$. The *(max-)degree of a graph* is the maximum among the degrees of its vertices.

A *self-loop* in a graph (V, E) is an edge $(v, v) \in E$. A set of *multiple edges* or *parallel edges* in a graph (V, E) is a set of edges connecting the same two vertices $v, w \in V$. A graph is *simple* if it does not contain either self-loops or multiple edges, otherwise it is called *multigraph*. In the following, unless otherwise specified, we always refer to simple graphs.

A graph is *directed* if its edges are *ordered* pairs of vertices. In a directed graph, an edge (v, w) is *oriented* from its *tail* (or *origin*) v to its *head* w; also, the edge is *outgoing* from v and *incoming* to w. The *indegree* of a vertex v is the number of its incoming incident edges; analogously, the *outdegree* is the number of its outgoing edges. A vertex whose indegree equals 0 is called *source*; analogously, a vertex whose outdegree equals 0 is called *sink*.

A graph $G'(V', E')$ is a *subgraph* of a graph $G(V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. A subgraph $G'(V', E')$ is *induced* by V' if, for each edge $(v, w) \in E$ such that $v, w \in V'$, $(v, w) \in E'$. A graph $G'(V', E')$ is a *spanning subgraph* of $G(V, E)$ if it is a subgraph of G and $V' = V$. A graph $G'(V', E')$ is a *supergraph* of a graph $G(V, E)$ if $V \subseteq V'$ and $E \subseteq E'$. A graph H is a *proper* subgraph (supergraph) of a graph G if G contains at least a vertex or an edge more (less) than H. A subgraph H of G is *maximal* under some condition *c* if there does not exists a graph H' fulfilling condition C and, at the same time is a subgraph of G and is a proper supergraph of H.

A graph is *complete* if, for every pair of vertices $u, v \in V$, edge $(u, v) \in E$. The complete graph on n vertices is denoted by K_n after Kuratowski, who first characterized planar graphs in [Kur30]. A graph is *bipartite* if it can be divided into two disjoint sets V_1 and V_2 such that no edge connects two vertices in the same set. A bipartite graph is complete if for each $v_i \in V_1$ and for each $v_j \in V_2$, edge $(v_i, v_j) \in E$. Complete bipartite graphs are denoted by $K_{a,b}$, where $a = |V_1|$ and $b = |V_2|$.

A *subdivision* of a graph G is a graph G' that can be obtained by replacing each edge of G with a sequence of new edges and new vertices such that S starts and terminates with an edge and contains an arbitrary number of new vertices. The *topological contraction* of an edge (v, w) consists of the replacement of v, w, and (v, w) with a single vertex u, of each edge (v, z) with an edge (u, z), and of each edge (w, t) with an edge (u, t). 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 topological contractions of edges.

A graph is *connected* if for any pair v, w of its vertices there exists a sequence of edges $e_1, \ldots, e_k$, with $k \geq 1$, such that:

- For each $i = 1, \ldots, k-1$ edges e_i and e_{i+1} have a common endvertex, and
- v is incident to e_1 and w is incident to e_k.

A graph that is not connected is said to be *disconnected*. In general, a graph that remains connected after the removal of any set of $k-1$ vertices is k-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, separating 2-sets, and separating 3-sets are also called *cutvertices*, *separation pairs*, and *separating triples*, 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 are shared by different blocks and the vertices belonging to a separation pair are shared by different separation pairs. Two blocks are *adjacent* if they share a cutvertex. Two adjacent blocks B_1 and B_2 are *consecutive* if there exists a pair of edges $(v, u) \in B_1$ and $(v, w) \in B_2$ such that (v, u) and (v, w) are consecutive in the rotation scheme of their shared cutvertex v, namely, vertices u and w are consecutive neighbors (see Section 1.2) of v.

1.2 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 (namely, with no self-intersections) curve between its *endpoints*, i.e., the points to which the end-vertices of the edge have been mapped. It is important to notice the difference between a graph, that is an abstract structure corresponding to a relationship among objects, and its drawing, that is a graphical representation of the graph.

A drawing is *planar* if no two edges intersect except, possibly, at their common end-points. A *planar graph* is a graph admitting a planar drawing. Planar graphs are probably the most studied class of graphs in Graph Theory, and surely the most studied class of graphs in Graph Drawing. In fact, a planar drawing of a graph provides extremely high readability of the combinatorial structure of the graph [PCJ97, Pur00]. See Figure 1.1 for a comparison between a non-planar and a planar drawing of the same graph.

A planar drawing of a graph induces a circular ordering, called the *rotation scheme* (or equivalently *combinatorial embedding*) of the edges incident to each vertex. A planar drawing, or equivalently a rotation scheme, of a graph partitions the plane into topologically connected regions called *faces*. A vertex (an edge) is *incident* to a face f if it belongs to sequence of vertices (edges) delimiting f. All the faces are bounded, except for one face, that we call *outer face* (or *external face*). The other faces are called

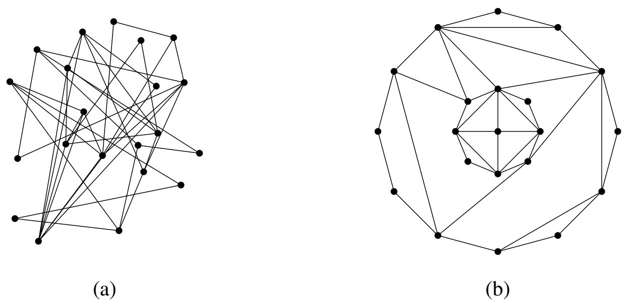


Figure 1.1: A non-planar (a) and a planar (b) drawing of the same graph.

internal faces. A vertex that is not incident to the extenral face is an *internal vertex*. Two planar drawings of a graph are *equivalent* if they induce the same rotation scheme and have the same outer face. A *planar embedding* of a graph is a class of equivalence of its drawings. A *plane graph* is a graph with a fixed planar embedding. All the planar drawings of a plane graphs are equivalent, while two drawings of the same graph in two different embeddings are not equivalent (see Figure 1.2). A combinatorial embedding is a class of equivalence of planar embeddings of a graph. Let G be a plane graph and let v be a vertex of G. Let also u and w be two neighbors of v in G. We say that u and w are *consecutive neighbors* of v in G if edges (v, u) and (v, w) are consecutive in the rotation scheme of v in G.

A plane graph is *maximal* (or equivalently is *internally triangulated*) if all its internal faces are delimited by cycles of three vertices. A planar graph is *maximal* (or, equivavalently, is a *triangulation*) if all its faces are bounded by cycles of three vertices. 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 and hence are often easier to deal with.

The *dual graph* of a combinatorially 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. Figure 1.3 shows an embedded planar graph and its dual graph. The dual graph of G only depends on the combinatorial embedding of G and not on the choice of the external face.

From the combinatorial and topological point of view, the first important result about planar graphs is the characterization given by Kuratowski [Kur30] in 1930,

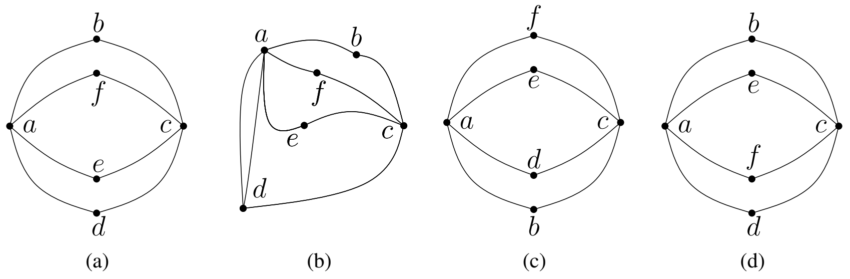


Figure 1.2: Four drawings of the same planar graph G. Drawings (a) and (b) are equivalent, as in both drawings the graph has the same combinatorial embedding and the same outer face; (c) has the same combinatorial embedding but a different external face with respect to the preceding drawings; and (d) depicts the graph in a different combinatorial embedding.

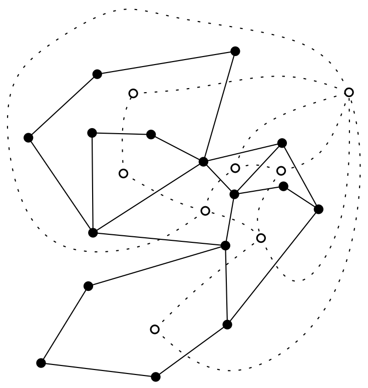


Figure 1.3: A planar graph, whose vertices are drawn as black disks and whose edges are drawn as solid segments, and its dual graph, whose vertices are represented by white circles and whose edges are represented by dashed lines.

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, dFOdM12, HT08]. Also, such testing algorithms can be suitably modified in order to compute planar embeddings if the test yields a positive result. 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 Families of Planar Graphs

In this section we characterize some notable subclasses of planar graphs we will deal with in the remainder of this thesis.

A *cycle* is a connected graph such that each vertex has degree exactly two (see Figure 1.4(a)). A *tree* is a connected acyclic (i.e., not containing any cycle) graph (see Figure 1.4(b)). 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) (see Figure 1.4(c)). A *leaf* of a tree is a vertex of degree one. A *leaf edge* is an edge incident to a leaf.

A *rooted tree* is a tree with one distinguished vertex, called *root*. In a rooted tree, the *depth* of a vertex 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 vertex has at most two (three) children. A tree is *ordered* if an order of the children of each vertex (i.e., a planar embedding) is specified. In an ordered binary tree we distinguish the *left* and the *right child* of a vertex. The *subtrees* of a vertex 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, a planar embedding in which all the vertices are incident to the outer face. An outerplanar graph, together with an outerplanar embedding is called an *outerplane graph*. An outerplanar graph is *maximal* if all bounded faces are delimited by cycles of three vertices. 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 Figure 1.5(a)). Also, outerplanar graphs have at most $2n - 3$ edges. Note that trees and cycles are outerplane graphs.

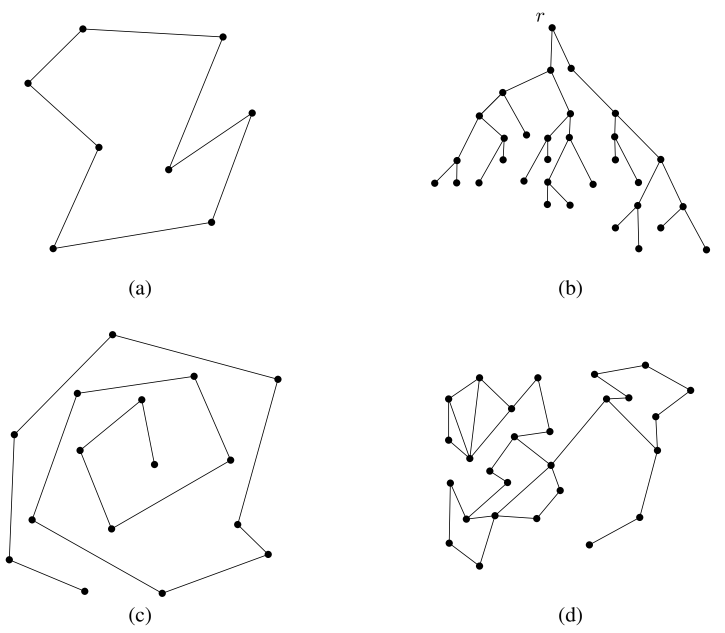


Figure 1.4: (a) A cycle. (b) A tree rooted at a vertex r. (c) A path. (d) An outerplane graph.

A *series-parallel graph* is inductively defined as follows. An edge (u, v) is 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 $G_1, \dots, G_k$ 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 $i = 1, \dots, k-1$ (see Figure 1.5(b)). A *parallel composition* of a set $G_1, \dots, G_k$ 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 Figure 1.5 (c)). From a combinatorial point of view, a series-parallel graph is a graph that contains no K_4-minor. If follows that any (connected) subgraph of a series-parallel graph is a series-parallel graph. Observe that outerplanar graphs are series-parallel graphs.

A graph G is a *k-tree* if it can be generated by a sequential addition of vertices (and their incident edges) in an order $v_1, \dots, v_n$ such that, for each $i > k$, vertex v_i has ex-

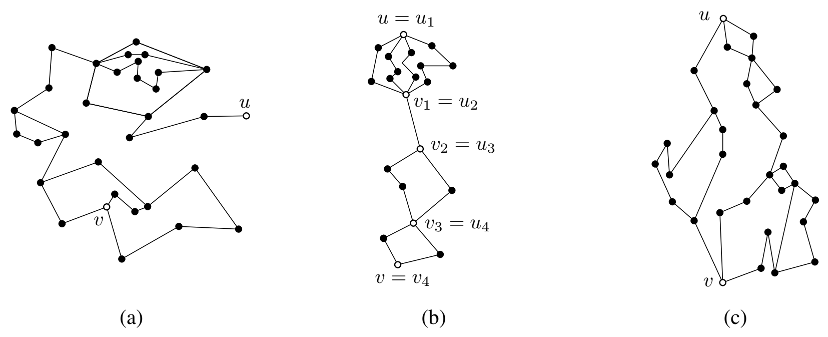


Figure 1.5: Figure 1.5(a) A series-parallel graph. Poles u and v are drawn as white circles. Figure 1.5(b) A series composition of a sequence of four series-parallel graphs. Figure 1.5(c) A parallel composition of a set $G_1, G_2, \ldots, G_k$ of series-parallel graphs.

actly k *predecessors* (i.e, neighbors with smaller index) and they form a *clique* (i.e, the subgraph of G induced by the predecessors of v_i is the complete graph K_k). A *partial k-tree* is a subgraph of a k-tree and have *treewidth* (i.e., the size of the largest clique in the graph) bounded by a constant. Partial k-trees received large attention since, as their treewidth is bounded by a constant, they allow for solving in polynomial time problems that are otherwise NP-hard ([AP89, CR05]). Trees coincide with 1-trees, while series-parallel graphs coincide with 2-trees. *Planar 3-trees*, also referred as *stacked triangulations* or *Apollonian graphs*, are special types of planar triangulations which can be generated from a triangle by a sequential addition of vertices of degree 3 inside faces. Namely, planar 3-trees can be inductively defined as follows:

- The complete graph K_3 on three vertices is a planar 3-tree.
- Let G be a planar 3-tree with n vertices and let a, b, and c be three vertices bounding a face of G. The graph G' obtained by adding vertex v and edges (a, v), (b, v), and (c, v) is a planar 3-tree with $n + 1$ vertices.

A graph G with $n \geq 4$ vertices is a *wheel* if it consists of a simple cycle $C = (v_1, \ldots, n_{n-1})$ on $(n - 1)$ vertices and the remaining vertex v is connected to all the vertices of C. Wheels are triconnected graphs. The complete graph on four vertices K_4 is a wheel on four vertices.

1.4 Drawing Conventions and Æsthetic Criteria

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 vertices makes the drawing aesthetically pleasant and easily readable by the human eye, and provides extremely high readability of the combinatorial structure of the graph, as confirmed by some cognitive experiments in graph visualization [PCJ97, Pur00, PCA02, WPCM02].

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.

In the following, we describe the most used drawing conventions and discuss some æsthetic criteria that characterize a good drawing of a graph.

1.4.1 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 Figures 1.6(a) and 1.6(c)) happen to be more readable than drawings in which edges are represented by poly-lines (known as *poly-line drawings*, see Figure 1.6(b)) or general curves, and drawings in which faces are drawn as convex polygons (known as *convex drawings*, see Figure 1.6(c)) are more readable than drawings in which this is not the case (see Figure 1.6). 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.

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

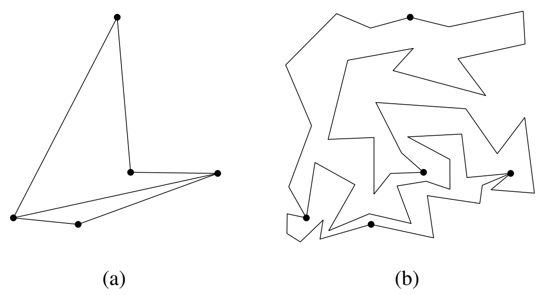


Figure 1.6: (a) A straight-line planar drawing of a planar graph G. (b) A poly-line planar drawing of G. (c) A convex drawing of G.

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 [Fár48], 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.

Other drawing conventions that we consider in this thesis are *monotone* and *orthogonal* drawings. A drawing of a graph is a *monotone drawing* if for every pair of vertices u and v there is a path drawn from u to v that is monotone in some direction. In other words, a drawing is monotone if, for any given direction d (e.g., from left to right) and for each pair of vertices u and v, there exists a suitable rotation of the drawing for which a path from u to v becomes monotone in the direction d. Monotone drawings will be discussed in depth in Chapter 9. In orthogonal drawing, edges are represented as poly-lines composed of horizontal and vertical segments. Orthogonal drawings will be discussed and extended in Chapter 10.

1.4.2 Æsthetic Criteria

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, FP08].

Chapter 2

Data Structures for Decomposing 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. In this chapter we present two such data structures, namely *block-butvertex trees* (bc-trees) (Section 2.1) and *SPQR-trees* (Section 2.2).

2.1 Block-Cutvertex Trees

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 [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. Figure 2.1 shows a connected planar graph and its block-cutvertex tree, rooted at a block B_1.

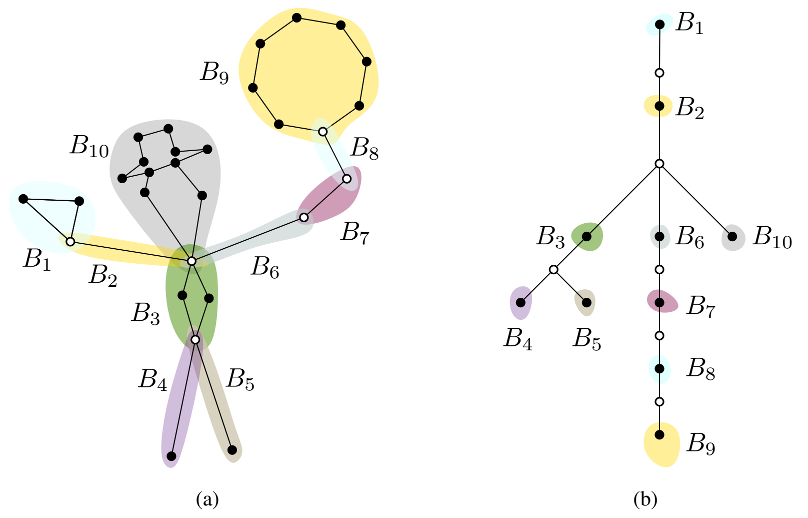


Figure 2.1: (a) A connected planar graph and (b) its block-cutvertex tree, rooted at B_1.

2.2 SPQR-trees

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 [DT90, DT96b, DT96a]. In this section 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.

In order to introduce SPQR-trees, we first give some definitions that will be useful in the following. 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 $\{u,v\}$.

Di Battista and Tamassia [DT96b] introduced SPQR-trees as rooted at one edge of G, called *reference edge*. However, SPQR-trees 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 only describe them as rooted trees.

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 $skel(\mu)$. The 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 of μ, is called the *virtual edge of* ν *in* $skel(\mu)$.

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 $pert(\mu)$.

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 are the *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 $skel(\mu)$ 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 $skel(\mu)$ is the graph obtained from G^* by replacing each subgraph G_i with the virtual edge e_i between u_i and v_i. The decomposition recurs on each G_i with μ as parent node and with $\{u_i, v_i\}$ as pair of vertices.

For each node μ of $\mathcal{T}_e$, we add to $skel(\mu)$ 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 $skel(\mu)$, 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''. Figure 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 $skel(\mu)$, or they belong to a set of virtual edges that induce a cycle in $skel(\mu)$.*

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 and an edge e of G, the SPQR-tree $\mathcal{T}_e$ of G with respect to e can be computed in linear time [GM01].

2.2.1 Drawing Plane Decomposed Graphs

In the following we give a description of how a straight-line planar drawing Γ, satisfying certain properties, of a plane (biconnected) graph G can be computed by exploiting its SPQR-tree T_e rooted at edge e.

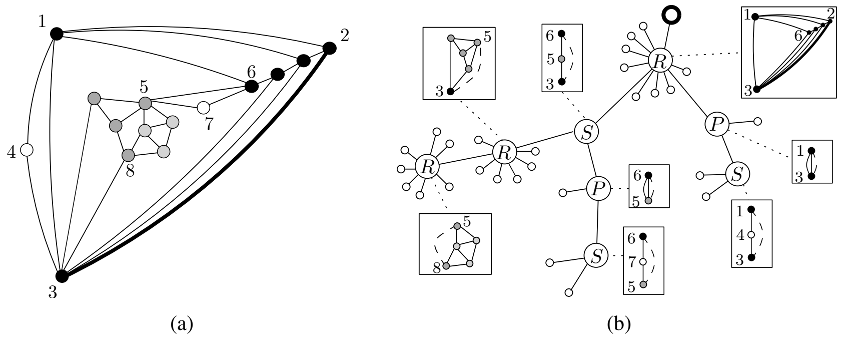


Figure 2.2: (a) A biconnected planar graph and (b) its SPQR-tree, rooted at any Q-node adjacent to the R-node whose internal vertices are black. The skeletons of the internal R-nodes of the tree are represented inside the boxes. The virtual edge representing the parent of a node μ in the skeleton of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

The drawing Γ is computed by assigning a region to each node μ of T_e and by suitably drawing the pertinent $pert(\mu)$ of μ inside such a region. The regions assigned to the nodes of T_e have been introduced in [ACD$^+$12] and [ADK$^+$13], and are of three types: Left boomerangs, right boomerangs, and diamonds. A *left boomerang* is a quadrilateral with vertices N, E, S, and W such that E is inside triangle $\triangle(N, S, W)$, where $|\overline{NE}| = |\overline{SE}|$ and $|\overline{NW}| = |\overline{SW}|$ (see Figure 2.3(a)). A *right boomerang* is defined symmetrically, with E playing the role of W, and vice versa (see Figure 2.3(b)). A *diamond* is a convex quadrilateral with vertices N, E, S, and W, where $|\overline{NW}| = |\overline{NE}| = |\overline{SW}| = |\overline{SE}|$. Observe that a diamond contains a left boomerang N_l, E_l, S_l, W_l and a right boomerang N_r, E_r, S_r, W_r, where $S = S_l = S_r$, $N = N_l = N_r$, $W = W_l$, and $E = E_r$ (see Figure 2.3(c)).

We assign boomerangs (either left or right, depending on the embedding of G) to S- and R-nodes, and diamonds to P- and Q-nodes. Drawing Γ is obtained by means of a top-down traversal of T_e (Figures 2.3(d) and 2.3(e)), as follows.

At the first step, consider the unique child μ of the root ρ of T_e and observe that, by construction, μ cannot be a Q-node. If μ is a P-node, assign a diamond to μ. Otherwise, μ is an R-node or an S-node. In this case, assign a boomerang (either left or right, depending on the embedding of G) to μ.

At each further step, consider a node μ of T_e and the region R (either a left/right boomerang or a diamond, depending on the type of μ and the embedding of G) assigned to it by the previous step of the algorithm. Construct a straight-line planar

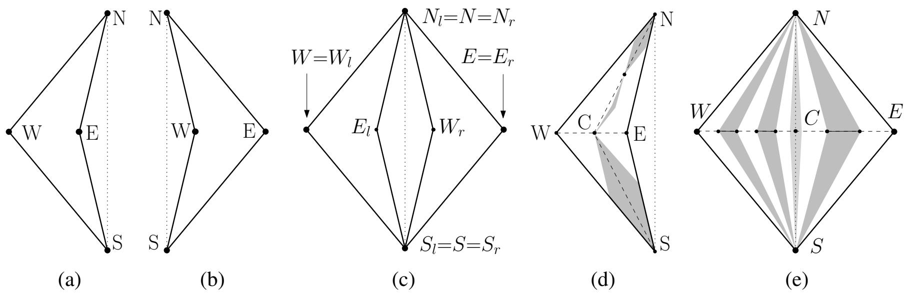


Figure 2.3: (a) A left boomerang. (b) A right boomerang. (c) A diamond. (d) Diamonds inside a boomerang. (e) Boomerangs (and a diamond) inside a diamond.

drawing of $skel(\mu)$ in a suitable fashion that depends on the required properties of Γ in the interior of R in such a way that the poles of μ lie on points N and S of R, according to the structure of G. Then, for each virtual edge $e_i = (u_i, v_i)$, corresponding to a child μ_i of μ in T_e, assign a region R_i (either a left/right boomerang or a diamond, depending on the type of μ_i and the embedding of G) to μ_i, in such a way that:

- u_i and v_i lie on points N_i and S_i of R_i;
- R_i is entirely contained in R; and
- R_i does not overlap (except possibly at points N_i and S_i) with any region R_j assigned to a child μ_j of μ, with $j \neq i$.

Then, recursively apply such a procedure to all the children of μ.

At the end of the recursive process, draw the edge e corresponding to the root ρ of T_e as a straight-line segment. Observe that, by construction and by the fact that e is the unique edge between its endvertices, the resulting drawing Γ is planar.

Part I

Morphing Planar Graph Drawings

Chapter 3

State of the Art

Given two straight-line planar drawings of a plane graph, a *planar morph* is a continuous transformation of the first drawing in which vertices move at uniform speed along straight-line trajectories and planarity is preserved at any time during the transformation.

In this chapter we give an overview of the main works about morphing planar graph drawings. We focus on Cairns’s result, since a similar recursive approach is applied in the algorithms described in Chapters 5 and 7.

3.1 Background

Even before that most of the well-known graph drawing concepts were formalized, the problem of proving whether two drawings of the same geometric object can be transformed one into the other without introducing any crossings arose among researchers.

To the best of our knowledge, this problem has first been studied by Tietze in 1914, who proved that two planar drawings of a polygon can be transformed into each other without introducing any crossings [Tie14]. Three years later, Smith [Smi17] simplified Tietze’s proof. Also, after Steinitz [Ste16] proved that every convex polyhedron forms a triconnected planar graph, and every triconnected planar graph can be represented as the graph of a convex polyhedron, Veblen [Veb17] and Alexander [Ale23] independently extended Tietze’s result to internally triconnected graphs.

In 1944 Cairns [Cai44a] focused on maximal plane graphs and gave the first algorithm for computing a morph of two straight-line planar drawings of such graphs by contracting a particular vertex and recursively computing a morph of the obtained (smaller) graph. Although Cairns’s algorithm requires a number of steps that is exponential in the size of the input graph, it is considered a milestone in the study of this problem as it had a great impact on the approaches used to tackle the problem in the years to come. A complete description of Cairns's algorithm is given in Sections 3.2 and 3.3.

Following this breakthrough, Bing and Starbird [BS78a, BS78b] and Ho [Ho73a, Ho73b, Ho74, Ho75], independently extended Cairns's result in several settings in which the outer face of the graph is delimited by a convex polygon which remains fixed durign the transformation. The main idea behind these papers is that any two drawings of the same plane graph can be augmented to represent two drawings of the same internally triconnected graph.

In 1983, Thomassen [Tho83] adopted a setting defined by Grünbaum and Shepard [GS81], i.e., in both drawings each face is delimited by a convex polygon, and, by using a contraction argument similar to Cairns's, he proved the existence of a morph in which such convexity is maintained at any time.

Unfortunately, both Cairns's and Thomassen's approaches require an exponential number of steps, as they exploit a double-recursion approach and do not take into account the trajectories of contracted vertices.

In order to find an effective approach for solving the problem in the general setting, researchers decided then to focus on simpler classes of graphs or to relax some constraints. Simple polygons and trees represented then a natural input. Kent, Carlson, and Parent [KCP92], Sederberg and Greenwood [SG92], and Guibas and Hershberger [GH94], proposed the firsts algorithms for computing morphs of polygon in a polynomial number of steps. These results have been later improved to $O(n \log n)$ in a paper by Hershberger and Suri [HS95], in which the authors prove that the same number of steps suffices also for binary trees. Other works on morphing simple polygons, some of them taking into account also additional constraints are [BSW97, NMWB08, AAD$^+$11].

Due to their structural properties, the next natural step was that of proving that a polynomial number of linear morphing steps suffices to transform drawings of plane orthogonal graphs, possibly maintaining edge orientations (as conjectured by Robinson [Rob81] considering a setting defined by Cairns [Cai44b]). The existence of a planar morph with such properties was already proved by Thomassen [Tho83]. In this setting, it has been proven [BLS05, LPS06, Spr07, BLPS13] that if there are more than two edge directions, then the problem of finding a morph preserving such directions is NP-hard, while for orthogonal drawings of plane graphs a polynomial number of steps is sufficient.

By allowing poly-line intermediate drawings, that is, "bending" the edges of the graph during the transformation, Lubiw and Petrick [LP11] found the first algorithm requiring a polynomial number of moves, namely $O(n^6)$, coping with general plane graphs.

Focusing back on the original setting, the first algorithms for computing planar linear morphs in $O(n)$ moves for simple classes of graphs as paths, outerplane graphs, and plane 3-trees have been proposed in [Ros10]. Following Cairns's approach, Alamdari *et al.* [AAC$^+$13] gave the first algorithm for morphing straight-line planar drawings of plane graphs that requires a polynomial number, namely $O(n^4)$, of moves. A few months later, Barrera-Cruz *et al.* [BHL13] gave a simpler technique for computing the motion of contracted vertices.

In this thesis we give an algorithm for morphing drawings of plane series-parallel graphs in a linear number of moves (see Chapter 5), show that $\Omega(n)$ moves are sometimes necessary (see Chapter 6), and improve to $O(n^2)$ the result by Alamdari *et al.* (see Chapter 7). After writing this thesis, we improved the upper bound on the number of moves to $O(n)$, thus obtaining an asymptotically optimal algorithm [ADD$^+$14].

This problem has been studied by Angelini *et al.* [ACDP08, ACDP13] also in a purely topological setting. Namely, the authors study how two planar embeddings of the same biconnected graph can be morphed one into the other while minimizing the number of elementary changes.

Due to its strict relation with computer graphics animation, this problem has been studied also in the setting in which vertices can move along non-linear trajectories. The most popular approach, in this setting, is based on Tutte's algorithm to construct planar drawings of plane triconnected graphs [Tut63], in which the vertices lying on the outer face induce a convex polygon and internal vertices are placed in the barycenter of the polygon induced by their neighbors. Floater and Gotsman [FG99] and Gotsman and Surazhsky [GS01] proved, for convex drawings of triconnected graphs, that expressing the position of each internal vertex as a convex combination of the positions of its neighbors and linearly interpolating the weights of this combination between the values computed in the two input drawings yields a planar morph. Surazhsky and Gotsman [SG01, SG03] later extended this approach to triangulations and plane graphs. In this setting the main target consists in minimizing the motion of vertices having the same position in the two drawings and, when applied in computer graphics, the distortion introduced in intermediate drawings, hence the function used to compute the weights that are interpolated during the animation has a relevant role. During the years, many types of coordinates for expressing the positions of internal vertices have been defined, some examples are [CGC$^+$02, MBLD02, Flo03, CdVPV03, FKR05, IMH05, JMD$^+$07].

Finally, some tools implementing graph-morphing algorithms have been presented in [FE02, EKP03, KL08].

3.2 Overview of Cairns's Algorithm

The algorithm proposed by Cairns [Cai44a] is a recursive process where the morph of an n-vertex maximal plane graph is reduced to two morphs of two suitable $(n-1)$-vertex maximal plane graphs, as follows.

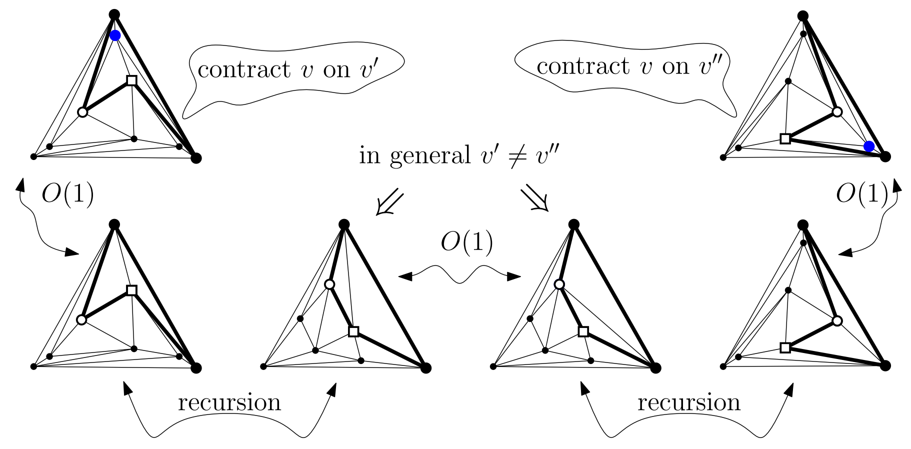


Figure 3.1: Overview of Cairns's Algorithm

Let Γ_s and Γ_t be two drawings of a maximal plane graph G with n vertices such that each of the three vertices on the outer face has the same position in both drawings, and let v be an internal vertex of G with at most five neighbors, see Figure 3.1.

First, v is removed from Γ_s and Γ_t and the resulting faces, which in general are not convex, are triangulated in Γ_s and Γ_t. This can be done by adding dummy edges incident to a neighbor v_s (v_t) of v lying on the boundary of the kernel of v in Γ_s (in Γ_t, respectively). Observe that, in general, $v_s \neq v_t$. Hence, two planar drawings Γ_σ and Γ_τ of two graphs G_σ and G_τ with $n-1$ vertices are obtained.

Second, a drawing Γ^v of G is constructed such that: (i) both v_s and v_t lie on the boundary of the kernel of v (for example, the polygon P induced by the neighbors of v in Γ^v is convex), and (ii) v lies in the centroid of its kernel. New drawings Γ^v_σ of G_σ and Γ^v_τ of G_τ are constructed from Γ^v by removing v and adding the missing edges inside the polygon P, which is always possible since both v_s and v_t have visibility on all the vertices of P in Γ^v, as (by construction) both v_s and v_t lie on the boundary of the kernel of v in Γ^v.

Cairns's algorithm is based on the observation that, starting from a morph in k

steps of G_σ from Γ_σ to Γ^v_σ and from a morph in h steps of G_τ from Γ^v_τ to Γ_τ computed inductively, it is possible to construct a pseudomorph in $k + h + 2$ steps of G from Γ_s to Γ_t, as follows.

The first step is used to "contract" v onto v_s. The subsequent k steps are the same as those of the morph of G_σ from Γ_σ to Γ^v_σ, with v moving in accordance with its neighbors.

Since both Γ^v_σ and Γ^v_τ have been obtained from the same drawing Γ^v, the neighbors of v have the same positions (and hence the cycle they induce is represented by the same polygon) in both drawings, vertex v can be ideally "uncontracted" from v_s and "contracted" with no additional morphing steps. In Section 3.3, we will show that the technique used to find a suitable position for v allows to reintroduce v in both Γ^v_σ and Γ^v_τ in such a way that it has the same position in both drawings.

Then, the subsequent h steps are the same as those of the morph of G_τ from Γ^v_τ to Γ_τ, with v moving in accordance to its neighbors. Finally, the last step is used to "uncontract" v from v_t and to place it in its position in Γ_t.

Actually, in order to avoid vertex overlapping during the morph, Cairns suggests to place v in the centroid of the kernel of the polygon induced by its neighbors (instead of collapsing it onto v_s and v_t) and to keep it in the centroid of such a varying kernel.

However, in Section 3.4 we show that this approach might lead to non-linear motion of some vertices during a single move, hence requiring several intermediate planar linear morphing steps.

3.3 Detailed Description of Cairns's Algorithm

This section is devoted to describe in depth Algorithm `Cairns_Morph`, that is the algorithm proposed by Cairns for morphing a planar straight-line drawing Γ_s of a maximal plane graph G into another planar straight-line drawing of G. We sketch Algorithm `Cairns_Morph` in the following.

Let Γ_s and Γ_t be two drawings of the same maximal plane graph G such that the vertices incident to the external face have the same positions in both drawings. Algorithm `Cairns_Morph` recursively computes a morph of Γ_s into Γ_t as follows.

Since, by hypothesis, the three vertices incident to the external face have the same positions in both drawings, if G has no internal vertices, then there is nothing to do.

So assume that G has at least an internal vertex. By Lemma 4.7, G has an internal vertex v such that $\deg(v) \leq 5$ (actually, in the lemma it is required that v is a candidate vertex, that is a stronger condition). Also, by Lemma 4.3, in any drawing Γ of G a neighbor x of v can be found such that x lies on the boundary of the kernel of v in Γ.

Algorithm Cairns_Morph

```
Require: $\Gamma_s$ and $\Gamma_t$ are two planar drawings of a maximal plane graph $G$;
    $\Gamma_s$ and $\Gamma_t$ have the same bounding polygon
    /* Initialize the morphing sequence */
 1. $M \leftarrow \emptyset$
    /* --------------------- Base Case ----------------------*/
 2. if $G$ has no internal vertices then
 3.     return $M$
    /* ------------------- Recursive Step -------------------*/
 4. $v \leftarrow$ internal vertex of $G$ such that $\deg(v) \leq 5$
 5. $v_s \leftarrow$ neighbor of $v$ on the boundary of the kernel of $v$ in $\Gamma_s$
 6. $v_t \leftarrow$ neighbor of $v$ on the boundary of the kernel of $v$ in $\Gamma_t$
 7. $\Gamma^v \leftarrow$ drawing of $G$ such that:
      $(i)$ $v_s$ and $v_t$ are in the kernel of $v$, and
     $(ii)$ $v$ lies in the centroid of its kernel
    /* Apply contractions */
 8. $G_\sigma = G/(v_s) \leftarrow$ contract $v$ onto $v_s$ in $G$
 9. $G_\tau = G/(v_t) \leftarrow$ contract $v$ onto $v_t$ in $G$
10. $\Gamma_\sigma = \Gamma_s/(v, v_s) \leftarrow$ contract $v$ onto $v_s$ in $\Gamma_s$ /* it's a drawing of $G_\sigma$ */
11. $\Gamma_\tau = \Gamma_t/(v, v_t) \leftarrow$ contract $v$ onto $v_s$ in $\Gamma_t$ /* it's a drawing of $G_\tau$ */
12. $\Gamma^v_\sigma = \Gamma^v/(v, v_s) \leftarrow$ contract $v$ onto $v_s$ in $\Gamma^v$ /* it's a drawing of $G_\sigma$ */
13. $\Gamma^v_\tau = \Gamma^v/(v, v_t) \leftarrow$ contract $v$ onto $v_s$ in $\Gamma^v$ /* it's a drawing of $G_\tau$ */
    /* Recursive calls */
14. $\mathcal{M}_\sigma \leftarrow$ Cairns_Morph($\Gamma_\sigma, \Gamma^v_\sigma$)
15. $\mathcal{M}_\tau \leftarrow$ Cairns_Morph($\Gamma^v_\tau, \Gamma_\tau$)
    /* Extend morphs $\mathcal{M}_\sigma$ and $\mathcal{M}_\tau$ to two morphs of $G$ */
16. $M_s \leftarrow$ remove contraction edges and reintroduce $v$ in each drawing of $\mathcal{M}_\sigma$
17. $M_t \leftarrow$ remove contraction edges and reintroduce $v$ in each drawing of $\mathcal{M}_\tau$ /*
    Construct the actual morph $M = \langle \Gamma_s, M_s, M_t, \Gamma_t \rangle$ */
18. $M \leftarrow$ append($M, \langle \Gamma_s \rangle$)          /* $M = \langle \Gamma_s \rangle$ */
19. $M \leftarrow$ append($M, M_s$)          /* $M = \langle \Gamma_s, \dots, \Gamma^v \rangle$ */
20. $M \leftarrow$ append($M, M_t$)
21. $M \leftarrow$ append($M, \langle \Gamma_t \rangle$)          /* $M = \langle \Gamma_s, \dots, \Gamma^v, \dots, \Gamma_t \rangle$ */
22. return $M$
```

Let then v_s and v_t be two of the neighbors of v lying on the boundary of the kernel of v in Γ_s and Γ_t, respectively (lines 5 and 6).

Let also Γ^v (line 7) be a drawing of G such that: (i) both v_s and v_t lie on the boundary of the kernel of v at the same time, e.g. the polygon P induced in Γ^v by the neighbors of v is convex; and (ii) v lies in the centroid of its kernel. Observe that P has not necessarily to be convex and that in some cases it cannot be drawn as a convex polygon, i.e. if an edge connecting two non-consecutive neighbors of v exists in G (see Figure 3.2).

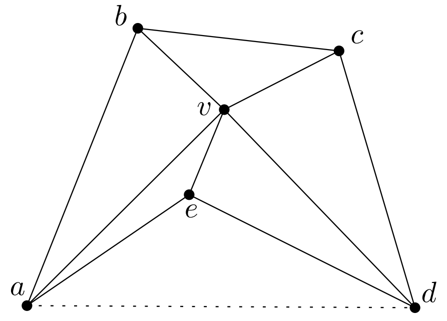


Figure 3.2: Polygon $\langle a, b, c, d, e \rangle$ does not admit a convex drawing because of the external chord (a, d), drawn as a dashed segment.

In such a case, Cairns suggests to draw P "as convex as possible". Namely, the polygon obtained from P together with its external chords is drawn convex (see Figure 3.3). Note that the vertices connected by external chords cannot, in any case, lie on the boundary of the kernel of v, hence vertex v will never be contracted onto such vertices.

Since v_s lies on the boundary of the kernel of v in both Γ_s and Γ^v, it can be contracted onto v_s. Let then Γ_σ and Γ^v_σ be the two drawings obtained by contracting v onto v_s in Γ_s and Γ^v, respectively. Analogously, let Γ_τ and Γ^v_τ be the two drawings obtained by contracting v onto v_t in Γ_t and Γ^v, respectively (see Figure 3.4).

Note that both Γ_σ and Γ^v_σ are drawings of the same graph $G_\sigma = G/(v, v_s)$, and the algorithm recursively computes a morph $\mathcal{M}_\sigma$ transforming Γ_σ into Γ^v_σ. Analogously, both Γ_τ and Γ^v_τ are drawings of the same graph $G_\tau = G/(v, v_t)$, and the algorithm recursively computes a morph $\mathcal{M}_\tau$ transforming Γ^v_τ into Γ_τ.

In order to obtain the final morph M transforming Γ_s into Γ_t, Cairns suggests to extend morphs $\mathcal{M}_\sigma$ and $\mathcal{M}_\tau$ into two morphs M_s and M_t of G by first removing from each drawing of the two morphs the edges added by the contraction of v onto v_s and

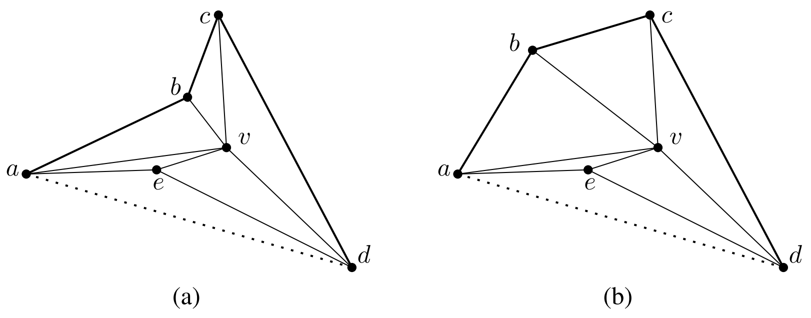


Figure 3.3: (a) Polygon $P = \langle a, b, c, d, e\rangle$ has an external chord (a, d). (b) A drawing "as convex as possible" of P: Polygon $P' = \langle a, b, c, d\rangle$, induced by P and its external chord (a, d), is convex. Due to the external chord (a, b), P does not admit any drawing in which a or d lie on the boundary of the kernel of v.

$$\begin{tikzcd}
\Gamma_s \arrow[dd] & & & & \Gamma_t \arrow[dd] \\
& & \Gamma^v \arrow[dl] \arrow[dr] & & \\
\Gamma_\sigma = \Gamma_s/(v, v_s) \arrow[r, "\text{recursion}"] & \Gamma^v_\sigma = \Gamma^v/(v, v_s) & & \Gamma^v_\tau = \Gamma^v/(v, v_t) \arrow[r, "\text{recursion}"] & \Gamma_\tau = \Gamma_t/(v, v_t)
\end{tikzcd}$$

Figure 3.4: Scheme of Algorithm `Cairns_Morph`.

onto v_t, respectively, and placing v in the centroid of its kernel in each drawing of the two morphs.

Recall that each contraction is actually realized by moving the "contracted" vertex v to the centroid of the polygon induced by its neighbors. By the "extension" technique described above, it follows that v lies in the centroid of its kernel in any drawing of both M_s and M_t. Observe that the first drawing of M_t and the last drawing of M_s are equal to Γ^v, as they are obtained by contracting v in such drawing. It follows that no additional morphing step is required to concatenate M_s and M_t in the final morph M.

Also, the contraction of v onto v_t and its subsequent replacement with the motion of v to the centroid of its kernel is actually used in the opposite sense during the morph. Namely, in the last drawing of M_t every vertex, with the exception of v, has the same position as the one it has in Γ_t. In fact, in the last drawing of M_s vertex v lies in the centroid of its neighbors. Analogously, Γ_s and the first drawing of M_s differ only for the position of v, which lies in the centroid of its kernel in the latter drawing.

Finally, morph $M = \langle \Gamma_s, \dots, \Gamma_t \rangle$ is obtained by the concatenation of Γ_s, M_s, M_t, and Γ_t.

3.4 Discussion

In this section we show that the morph computed by Algorithm `Cairns_Morph` needs an exponential number of steps and that some vertices may move along non-straight-line segments during such a morph.

3.4.1 Total Number of Moves

Let $T(n)$ be the number of steps that compose M. The first and the last step of M move v from its initial position to the centroid of the kernel of the polygon P_s induced by its neighbors in Γ_s and from the centroid of its kernel in Γ_t to its final position, respectively. Such operations are realized with a single morphing step each. The remaining part of the morph is computed recursively on two smaller graphs and then extended to two morphs M_s and M_t of G with no additional morphing steps. Each of M_s and M_t needs $T(n-1)$ steps. The total number of steps needed by M is then $T(n) = 2T(n-1) + 2$, which gives $T(n) = \Theta(2^n)$.

3.4.2 Possible Non-linear Trajectories

In order to avoid vertex overlapping during the morph, Cairns suggests to place v in the centroid of its kernel (instead of collapsing it onto v_s and v_t) and to keep it in the centroid of such a varying kernel. However, we observe that:

(i) since the corners of the kernel might move along curves that are not straight-line, this could result in a non-linear movement of v (see Figure 3.5);

(ii) it might happen that the number of the corners of the kernel changes during the morph (see Figure 3.6 for an example). This implies that, when the number of corners of the kernel of v changes, the position of the centroid "jumps" (namely, changes without continuity) from a point to another, thus originating a non-linear morph.

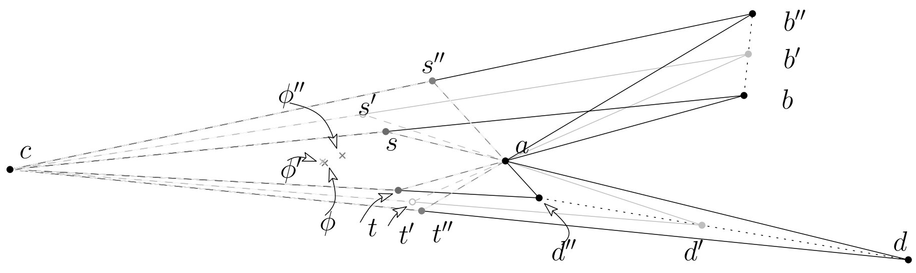


Figure 3.5: A linear morphing of polygon (a, b, c, d), with kernel a, t, c, s, snapshotted in three instants. Corner s of a, t, c, s moves along a non-straight-line trajectory. As a consequence, the centroid ϕ of a, t, c, s does not move along a straight-line.

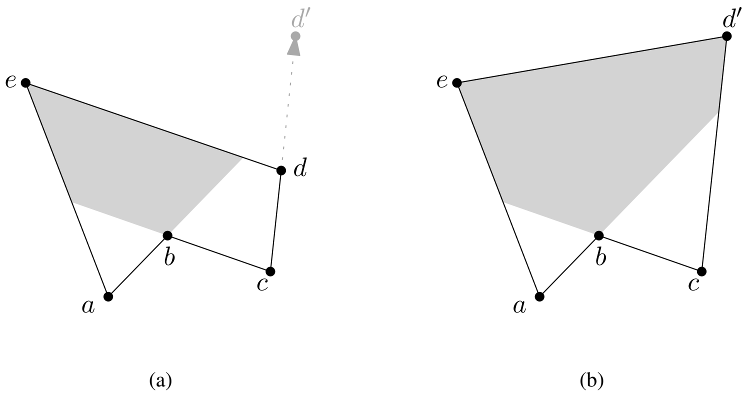


Figure 3.6: The number of the vertices bounding the kernel of pentagon $\langle abcde \rangle$ changes during a single linear morphing step.

Chapter 4

Geometric and Topological Tools for Morphing Planar Graphs

In this chapter we give some definitions, tools and techniques that will be used to compute morphs of planar graph drawings. The chapter is structured as follows. In Section 4.1 we give some preliminary definitions about morphs and pseudo-morphs of planar graph drawings. In Section 4.2 we characterize vertices that are "candidate" to the contraction, which, in some cases, we use for reducing the size of the problem. In Section 4.3 we describe how a drawing of a simply connected graph can be augmented to a drawing of a biconnected graph by satisfying certain properties. In Section 4.5 we describe a simple technique to morph two drawings of a triangle, while in Section 4.4 we introduce convex coordinates, which can be exploited to express the position of points lying in the interior of a convex polygon with respect to its vertices. Finally, in Section 4.6 we describe how a pseudo-morph of a plane graph can be extended to an actual morph of the same graph.

4.1 Definitions

A *(planar linear) morphing step* $\langle \Gamma_1, \Gamma_2 \rangle$, also referred to as *linear morph* or *step*, of two straight-line planar drawings Γ_1 and Γ_2 of a plane graph G is a continuous transformation of Γ_1 into Γ_2 such that:

- all the vertices simultaneously start moving from their positions in Γ_1;
- each vertex moves at constant speed along a straight-line trajectory;
- all the vertices simultaneously stop at their positions in Γ_2;

- no crossing occurs between any two edges during the transformation; and
- no two vertices are mapped to the same point during the transformation.

A *morph* $M = \langle \Gamma_s, \dots, \Gamma_t \rangle$ of two straight-line planar drawings Γ_s into Γ_t of a plane graph G is a finite sequence of morphing steps that transforms Γ_s into Γ_t. A morph is *unidirectional* if, at each step, all the vertices move along parallel straight-line trajectories. Let H be any subgraph of G and let Ψ_s and Ψ_t be two drawings of H obtained by restricting Γ_s and Γ_t, respectively, to the vertices and the edges of H. Than a morph $M_H = \langle \Psi_s, \dots, \Psi_t \rangle$ can be obtained by restricting each intermediate drawing of M to the vertices and the edges of H.

Let Γ be a planar straight-line drawing of a plane graph G. The *kernel* of a vertex v of G in Γ is the open convex region of the plane such that: for each point p of the region, placing v onto p while maintaining unchanged the position of any other vertex of G yields a planar straight-line drawing of G. See Figure 4.1(a). Note that: (i) the kernel of v in Γ is non-empty; (ii) v is the unique vertex of G lying in the interior of its kernel; and (iii) the boundary of the kernel of v might not contain, in general, any of the neighbors of v.

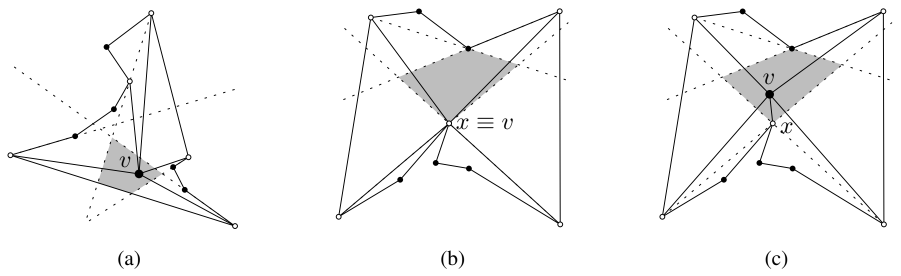


Figure 4.1: (a) Kernel of a vertex v in a planar straight-line drawing of G. (b) Uncontraction kernel of v in a planar straight-line drawing Γ' of $G' = G/(v, x)$. (c) Kernel of v in a drawing Γ^* of G obtained by uncontracting v from x in Γ'.

If a neighbor x of v lies on the boundary of the kernel of v in Γ, we say that v is *x-contractible*. Further, we define the *contraction of v onto x* in Γ as the operation resulting in:

(i) a simple plane graph $G' = G/(v, x)$ obtained from G by removing v and by replacing each edge (v, w), where $w \neq x$, with an *inherited* edge (x, w) (possible copies of the same edge are removed); and

(ii) a planar straight-line drawing Γ' of G' such that each vertex different from v is mapped to the same point as in Γ.

Note that, by the convexity of the kernel of v, the straight-line segment representing edge (v, x) entirely lies in the kernel of v, except for its endpoint x. It follows that no crossing occurs if v moves towards x along this segment.

Consider a drawing Γ'' of $G' = G/(v, x)$. We define the *uncontraction of v from x* in Γ'' as the operation resulting in a planar straight-line drawing Γ^* of G such that each vertex of G' has in Γ^* the same position as in Γ''. Further, we define the *uncontraction kernel* of v in Γ'' as the open (convex) region of the plane where v can be placed when performing the uncontraction from x. See Figure 4.1(b).

In Lemma 4.1 we prove that the uncontraction kernel of v in Γ'' is always non-empty and coincides with the kernel of v in Γ^* (see Figure 4.1(c)).

Let y and z be two vertices of G such that y and v, and z and v are consecutive neighbors of x in G (see Figure 4.2(a)). In $G' = G/(v, x)$ the inherited edges (x, w):

- appear between edges (x, y) and (x, z), namely, replace edge (x, v) of G in G';
- are consecutive in the rotation scheme of x in G'; and
- appear in the same circular order as edges (v, w) appear in G (see Figure 4.2(b)).

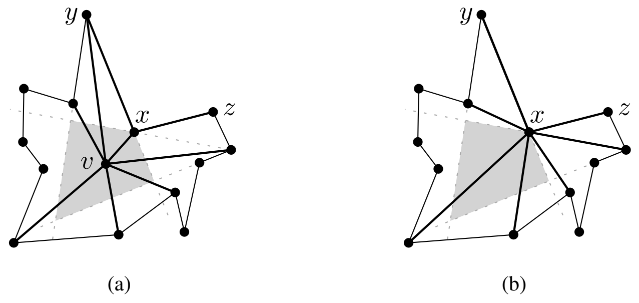


Figure 4.2: (a) Since vertex x lies on the boundary of the kernel of v (grey region), v is x-contractible. Vertices y, v, and z are consecutive neighbors of x. (b) After the contraction of v onto x, the inherited edges of x replace edge (v, x) in the rotation scheme of x.

Also, since vertex x is adjacent in G' to all the neighbors of v in G, x lies on the boundary of the uncontraction kernel of v in Γ'', and hence on the boundary of the kernel of v in Γ^* (grey regions in Figure 4.2).

Lemma 4.1 *Let v be a vertex of G and let x be a neighbor of v. Further, let Γ be a straight-line planar drawing of G such that v is x-contractible. Then x lies on the boundary of the uncontraction kernel of v in any straight-line planar drawing Γ' of $G' = G/(v, x)$.*

Proof: Suppose for a contradiction that x does not lie on the boundary of the uncontraction kernel of v in Γ' and observe that, since Γ' is planar, the edges that x inherited from v do not cross any other edge of G' and appear in the same circular order around x as they appeared around v, except for the absence of edge (v, x).

As in the proof of Fáry's Theorem [Fár48], there exists an $\epsilon > 0$ such that the disk D with radius ϵ centered at x is entirely contained in the kernel of x in Γ'.

Observe that the uncontraction kernel of v in Γ' must be contained in the face f obtained by removing the inherited edges of x from G'. Also, note that the intersection between f and D is non-empty and defines a sector S of D.

It follows that the crossings occurring in the straight-line drawing Γ'' obtained by uncontracting v from x to any point of S while maintaining any other vertex at its position in Γ' involve only edges incident to v. This is impossible. □

In order to perform contractions and uncontractions, we focus on low-degree "candidate" vertices, leveraging on their topological and geometric properties. Namely, we say that a vertex v is a *candidate to the contraction* (or simply a *candidate vertex*) if:

(i) $\deg(v) \leq 5$; and

(ii) if two of its neighbors, u and w, are connected by an edge, then (u, v, w) is a simple face of G.

Let v be a candidate vertex of G (the existence of a candidate vertex in every plane graph is proved in Section 4.2) and assume that, if v has at least three neighbors, they induce a cycle C in G. Observe that this is not a loss of generality because for any two consecutive neighbors u and w of v, there exists a face having v, u, and w on its boundary and hence the introduction of edge (u, w) yields a planar supergraph of G.

In the following lemmata we prove that, under this assumption, v is contractible on at least one of its neighbors. Let Γ be a straight-line planar drawing of G and let x be a vertex of C.

In order to simplify the proofs, we first show that if each $\overline{xv_i}$, with v_i neighbor of v and $v_i \neq x$, do not intersect any edge that is not incident to v, then x lies on the boundary of the kernel of v, i.e., vertex v is x-contractible.

Lemma 4.2 *Let v be a candidate vertex of G such that the closed polygonal region delimited by its neighbors does not contain any vertex other than v and its neighbors.*

Let also Γ *be a straight-line planar drawing of* G *and let* x *be a neighbor of* G *such that, for any neighbor* v_i *of* v *different from* x*, if segment* $\overline{xv_i}$ *crosses an edge in* Γ*, it is incident to* v*. Then,* v *is* x*-contractible.*

Proof: Augment Γ and G with an edge between any pair of consecutive neighbors of v. Observe that, after this augmentation, Γ is still planar, since (by hypothesis) no vertex of G lies along the straight-line segment representing an edge between two consecutive neighbors of v. By definition, the kernel of v is an open convex region of the plane. Hence, it suffices to prove that any internal point p of the segment $\overline{vx}$ representing edge (v,x) in Γ belongs to the kernel of v in Γ.

Suppose, for a contradiction, that it does not. Let then Γ' be the non-planar straight-line drawing of G obtained from Γ by placing v at some point p lying along $\overline{vx}$ in Γ and maintaining fixed the positions of all the remaining vertices of G.

If $\deg(v) = 1$, the only possible crossing must involve edge (v,x) as it is the unique edge whose drawing has changed. Hence, some edges or some vertices lie along or traverse some point along segment $\overline{vx}$ in Γ', implying the fact that Γ is planar. This is a contradiction.

If $\deg(v) = 2$, denote by u and w be the two neighbors of v and observe that, by hypothesis, triangle $\Delta = uvw$ delimits a face of G in Γ. Assume $x = u$. Since (i) v is the only vertex of G having a position in Γ' that is different from the one in Γ; (ii) Γ is planar; and (iii) p lies along the segment representing edge $(v,u) = (v,x)$ in Γ, the crossings occurring in Γ' must involve edge (v,w). Also, since the triangle Δ' delimited by u, v, and w in Γ' is entirely contained in Δ, the endpoints of the edges crossed by (v,w) lie inside Δ in Γ. This is a contradiction to the fact that v is candidate vertex, as face (uvw) would not be empty. Observe that the same argument applies to w, hence v can be contracted also onto w.

Analogously, if $3 \leq \deg(v) \leq 5$, the crossings occurring in Γ' involve at least an edge incident to v and at least an edge with both endpoints lying inside one of the faces delimited by v and a pair of its consecutive neighbors. This contradicts the fact that v is a candidate vertex. □

In the following we prove that, if v is a candidate vertex and each pair of its consecutive neighbors are connected by an edge, then v has a neighbor x lying on the boundary of its kernel, that is, v is x-contractible.

Lemma 4.3 *Let* G *be a plane graph and let* Γ *any straight-line planar drawing* G*. Further, let* v *be a candidate vertex of* G *such that each pair of consecutive neighbors of* v *is connected by an edge. Then, there exists a neighbor* x *of* v *such that* v *is* x*-contractible in* Γ*. Moreover,* (i) *if* $1 \leq \deg(v) \leq 3$*, vertex* v *can be contracted on any* *of its neighbors, and* (ii) *if* $\deg(v) = 4$*, vertex* v *can be contracted also on its unique neighbor that is not consecutive to* x*.*

Proof: Augment Γ and G with an edge between any pair of consecutive neighbors of v. Observe that, after this augmentation, Γ is still planar, since (by hypothesis) no vertex of G lies along the straight-line segment representing an edge between two consecutive neighbors of v. If $1 \leq \deg(v) \leq 3$, let x be any of the neighbors of v. Since for any neighbor $y \neq x$ of v edge (y, x) belongs to G, by Lemma 4.2, v is contractible on any of its neighbors. So assume that $4 \leq \deg(v) \leq 5$. Let C be the cycle induced by the neighbors of v in G. Let also G' and Γ' be the graph and its drawing obtained by removing v and its incident edges from Γ and G, respectively. Consider the polygon P induced by C in Γ'. Observe that, as a consequence of Meisters's Two Ears Theorem [Mei75], polygon P can be subdivided into $|P| - 2 = \deg(v) - 2$ triangles by adding edges only, and that such these triangles have a common vertex, say x. Finally, observe that, if $\deg(v) = 4$, the two triangles into which P has been subdivided have two common vertices, both connected to the two remaining neighbors of v. By Lemma 4.2, the statement follows. □

Let Γ_1 and Γ_2 be two drawings of the same plane graph G in which v is contractible onto the same neighbor x. We define a *pseudo-morph* of Γ_1 into Γ_2 as a sequence of operations composed of:

(i) the contraction of v onto x in Γ_1, resulting in a drawing Γ'_1 of $G' = G/(v, x)$;

(ii) the morph of Γ'_1 into a drawing Γ'_2 of G'; and

(iii) the uncontraction of v in Γ'_2 resulting in Γ_2.

4.2 Candidate Vertices in Plane Graphs

In this section[1] we prove that any plane graph contains a candidate vertex with certain properties.

Lemma 4.4 *Let* Γ *be a straight-line planar drawing of a maximal outerplane graph* G *with at least three vertices. Then, there exists a face* $f = (u, v, w)$ *of* G *such that:* (i) v *has degree* 2 *in* G*; and* (ii) v *is both* u*- and* w*-contractible in* Γ*. Moreover, if* G *has only two vertices with degree* 2*, then:* (iii) u *has degree* 3*; and* (iv) u *is* w*-contractible in* Γ*.*

[1]Part of the contents of this section are joint work with Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani, and have appeared in [AFPR13].

Proof: We first prove that v exists in G. Observe that if G has only three vertices, then, by the maximality of G, all its vertices have degree 2 and cycle (u, v, w) bounds the unique internal face of G. So, assume that G has more than three vertices and observe that Γ can be viewed as an internally-triangulated polygon. Since, by Meisters's *Two Ears Theorem* [Mei75], each polygon has at least two *ears* (i.e., vertices with degree 2), v is an ear of Γ, and hence a degree-2 vertex of G.

Let then u and w be the two neighbors of v in G. Since G is a maximal outerplane graph, edge (u, w) belongs to G and cycle $C = (u, v, w)$ delimits a face f of G. By the maximality of G and the planarity of Γ, the triangle Δ representing C and bounding f is empty, as no vertex other than u, v, and w, and no edge different from those in C is incident to f. By Lemma 4.3, v is both u-contractible and w-contractible.

Assume that, of all the at least four vertices of G, only one (say t), other than v has degree 2. In the following we prove conditions (iii) and (iv). Let G' and Γ' be the graph and its drawings obtained by removing s, t, and their incident edges from G and Γ, respectively. Note that G' is a maximal outerplane graph and hence has at least two vertices with degree 2, say x and y. Since all the vertices of G, different from the neighbors of v and t, have the same degree in both G and G', vertices x and y must be neighbors of v and t in G. Also, since G is a maximal outerplane graph, x and y cannot be both neighbors of v or both neighbors of t. It follows that one of the neighbors of v, say u, has degree 3 in G.

Finally, we prove that u is w-contractible. Since G is a maximal outerplanar graph, vertices u and w have a common neighbor different from v, say z. It follows that w is connected to all the three neighbors of u. Also, since Γ is planar and no vertex, other then u, w, and z is incident to face (u, w, z), vertex w lies on the boundary of the kernel of u. Hence, u is w-contractible. □

Lemma 4.5 *Let Γ be a planar straight-line drawing of a biconnected outerplane graph G with n vertices. There exists a degree-2 vertex v of G that is contractible on both its neighbors in Γ and the contraction yields an outerplane graph with $n-1$ vertices.*

Proof: Observe that Γ can be augmented to a planar straight-line drawing Γ^+ of a maximal outerplane graph G^+ by triangulating each internal face, e.g., by applying Chazelle's algorithm [Cha91]. Then, by Lemma 4.4, the statement follows. □

In the following we prove that any plane graph contains a candidate vertex. Also, if the graph is maximal planar, the candidate vertex can be chosen to be internal.

Lemma 4.6 *Let $G = (V, E)$ be a plane graph and let $G' = (V, E')$, with $E' \subseteq E$, be any connected subgraph of G. If v is a candidate vertex of G, then it is also a candidate vertex of G'.*

Proof: We prove the statement for the case in which $E \setminus E' = \{e\}$. Assume for a contradiction that v is not a candidate vertex of $G' = G - e$. Since the removal of e does not increase the degree of v, there exists a 3-cycle (u,v,w) that does not bound a simple face of G'. As the reintroduction of e cannot transform the 3-cycle (u,v,w) into a simple face, v cannot be a candidate vertex of G, a contradiction. □

Lemma 4.7 *Every plane graph contains a candidate vertex v.*

Proof: Let Γ be a planar straight-line drawing of a plane graph $G = (V, E)$. Add vertices a, b, and c so that the triangle composed by these vertices completely encloses Γ. Further, augment the obtained drawing to the straight-line drawing Γ' of a maximal plane graph G' by adding dummy edges only [Cha91]. In the following we show that G' contains a candidate vertex v different from a, b, and c. Observe that, by Lemma 4.6, it suffices to prove this statement. By the identity $\sum_{p \in V} \deg(p) = 6n - 12$, there is at least one non-boundary vertex v' of G' whose degree is at most 5. If no edge of G' connects two non-consecutive neighbors of v', then $v = v'$ is a candidate vertex of G'. Otherwise, let u and w be two non-consecutive neighbors of v' connected by an edge in G'. Observe that there exist two neighbors of v' such that one of them lies inside triangle $\langle v', u, w \rangle$ and the other one lies outside it. Consider the sub-drawing Γ'' of Γ' obtained by removing all the vertices and the edges of G' lying outside triangle $\langle v', u, w \rangle$. Since G' is a maximal plane graph, Γ'' is a drawing of a maximal plane subgraph G'' of G' having at least one vertex less than G'. Hence, there exists a non-boundary vertex v'' of G'' whose degree is at most 5. By repeatedly applying this argument, we eventually find a candidate vertex $v = v^*$ of some subgraph G^* of G' (and hence, by Lemma 4.6, of G). □

4.3 Merging Two Consecutive Blocks of a Simply-Connected Graph

In the following we describe a technique to augment a simply-connected graph G to a biconnected graph G' which satisfies certain properties. This technique will be applied in Chapter 5 in order to compute morphs of plane series-parallel graphs. Namely, we prove that:

(a) if G is a plane series-parallel graph, then G' is a plane series-parallel graph;

(b) if G is outerplane, the resulting graph G' is outerplane; and

(c) given any straight-line planar drawing Γ of G, there exists a planar straight-line drawing Γ' of the resulting graph G' such that each vertex of G has the same positions in Γ and in Γ'.

So, le v be a cutvertex of G and let B_1 and B_2 be two blocks of G such that there are edges $(v, u) \in B_1$ and $(v, w) \in B_2$ that are consecutive in the counter-clockwise order of the edges around v in G, see Figure 4.3(a). Graph G' is obtained by adding a new vertex z and edges (u, z) and (w, z). In the following lemma, we prove that G' satisfies conditions (a), (b), and (c).

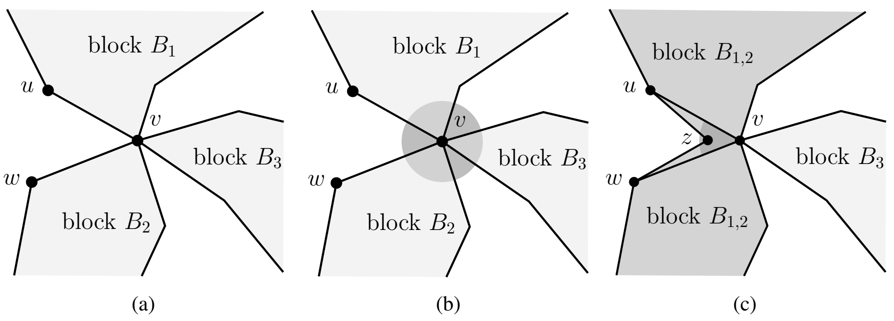


Figure 4.3: (a) A cut-vertex v of a connected outerplane graph G. (b) Constructing disk D centered at v. (c) Drawing Γ' is obtained by placing z in S (dark grey wedge).

Lemma 4.8 *Let v be a cutvertex of a simply-connected graph G and let B_1 and B_2 be two consecutive blocks of G around v with edges $(v, u) \in B_1$ and $(v, w) \in B_2$. The graph G' obtained from G by adding a vertex z and edges (u, z) and (w, z) satisfies conditions (a), (b), and (c).*

Proof: We first prove that G' satisfies condition (a), i.e., if G is a plane series-parallel graph, then G' is a plane series-parallel graph.

Suppose, for a contradiction, that G' is not a plane series-parallel graph. It follows that G' contains a subdivision of the complete graph on four vertices K_4, i.e., there is a set V_{K_4} of four vertices of G' such that any two of them are joined by three vertex-disjoint paths. Observe that the vertices in V_{K_4} cannot belong to different blocks of G'. Further, since G is a plane series-parallel graph, the vertices in V_{K_4} belong to $B_{1,2}$. Since z has degree two, $z \notin V_{K_4}$; hence, the vertices in V_{K_4} are also vertices of G. This gives a contradiction since: (i) The vertices in V_{K_4} cannot all belong to B_1, as otherwise G would not be series-parallel, contradicting the hypothesis; (ii) for the same reason, the vertices in v_{K_4} cannot belong to B_2; and (iii) the vertices in V_{K_4} cannot belong both to B_1 and B_2, as otherwise there could not exist three vertex-disjoint paths joining them in G', contradicting the hypothesis that G' contains a subdivision of K_4. Since outerplane graphs are also plane series-parallel graphs, in order to prove that G' satisfies condition (b), it suffices to show that all the vertices of G' are incident to the external face of G'. Since G is outerplane, v, u, and w lie on the external face. Also, since edges (v, u) and (v, w) are consecutive in the rotation scheme of v in G, no vertex of G' (other than u, v, w, and z) is incident to (and hence enclosed in) the face formed by the addition of edges (u, z) and (w, z). Then, all the vertices of G are incident to the external face of G'. Finally, since z can be added to any face of G shared by u, v, and w, it can be added to the external face of G, that is, the external face of G'.

We conclude the proof showing that G' satisfies condition (c), namely that any planar straight-line drawing of G can be augmented to a planar straight-line drawing of G'. Let Γ be a planar straight-line drawing of G. As shown in the proof of Fáry’s Theorem [Fár48], there exists an $\epsilon > 0$ such that the disk D with radius ϵ centered at v lies entirely in the kernel of v in Γ. Also, moving vertex v to any point of D while maintaining unchanged the positions of the other vertices of G results in a planar straight-line drawing of G. Let f be the face of G, shared by u, v, and w, in which z has to be placed. The set S of feasible points to place z is defined as the set of points with direct visibility on vertices u and w in the intersection between f and D. Drawing Γ' is hence constructed from Γ by placing z at any point of S (see Figure 4.3(c)). □

4.4 Convex Coordinates

Let P $(v_1, \dots, v_k)$ be a convex polygon with k vertices. Any point p lying in the interior of P can be expressed as a *convex combination* of the vertices of P. Namely, there exist coefficients $\lambda_1, \dots, \lambda_k$ such that $0 < \lambda_i < 1$, for each $1 \leq i \leq k$; and $\sum_{i=1}^{k} = 1\lambda_i$. For each $i = 1, \dots, k$, denote by (x_i, y_i) the coordinates of vertex v_i.

Let (x_i, y_i) be the coordinates of the vertices of P. The coordinates x_p and y_p of point p can be expressed as:

$$x_p = \sum_{i=1}^{k} \lambda_i x_i \tag{4.1}$$

$$y_p = \sum_{i=1}^{k} \lambda_i y_i . \tag{4.2}$$

Coefficients $\lambda_1, \dots, \lambda_k$ are called the *convex coordinates* of p in P.

Lemma 4.9 *Let G be a plane graph with n vertices, whose external face is bounded by cycle $S = (v_1, \dots, v_k)$. Let also Γ_1 and Γ_2 be two drawings of G such that:*

- *cycle S induces convex polygons Σ_1 and Σ_2 in Γ_1 and Γ_2, respectively;*
- *each internal vertex v_i, $k+1 \leq i \leq n$ has the same convex coordinates in both Γ_1 and Γ_2; and*
- *the planar linear morph $M_\Sigma = \langle \Sigma_1, \Sigma_2 \rangle$ is such that polygon $\Sigma(t)$, induced by S at any time instant t, $0 \leq t \leq 1$, of M_Σ is convex.*

Then, morph M_Σ can be extended to a planar linear morph $M = \langle \Gamma_1, \Gamma_2 \rangle$ of G such that:

- *each external vertex v_j, $1 \leq j \leq k$, moves as in M_Σ; and*
- *the convex coordinates of each internal vertex v_i, $k+1 \leq i \leq n$, with respect to the positions of the vertices of S remain fixed throughout M.*

Proof: For each vertex v_j, $1 \leq j \leq k$, and for each v_i, $k+1 \leq i \leq n$, denote by $v_i(t)$ and $v_j(t)$, respectively, the position of vertex v_i and vertex v_j at any time instant t, $0 \leq t \leq 1$, during M.

Observe that, when $t = 1$ ($t = 2$), $v_i(t)$ and $v_j(t)$ identify the initial (final) position of v_i and v_j in Γ_1 (Γ_2). Also, by hypothesis, for each $j = 1, \dots, k$ and for each t, $0 \leq t \leq 1$, $v_j(t)$ is the same in both M and M_Σ, while for each $i = k{+}1, \dots, n$ the convex coordinates of v_i remain fixed during M.

We first show that morph M is linear. Since, by hypothesis, boundary vertices move along straight-line trajectories, it suffices to show that each internal vertex v_i moves along the straight-line segment connecting its position in Γ_1 to its position in Γ_2, namely, for any $0 \leq t \leq 1$:

$$v_i(t) = (1-t)v_i(0) + t\, v_i(1). \tag{4.3}$$

By hypothesis, at any time instant, the position $v_i(t)$ of any internal vertex v_i is expressed as a fixed convex combination with coefficients $\lambda_{ij}$of the positions of the vertices v_j of S, namely:

$$v_i(t) = \sum_{j=1}^{k} \lambda_{ij} v_j(t). \tag{4.4}$$

By expanding the summation in Equation (4.4) we obtain Equation 4.5 and hence Equation 4.6, which can be rewritten as Equation 4.7, thus proving that v_i moves along the straight-line segment connecting its positions in Γ_1 and Γ_2.

$$v_i(t) = \lambda_{i1}v_1(t) + \lambda_{i2}v_2(t) + \ldots \lambda_{ik}v_k(t) \tag{4.5}$$

$$v_i(t) = \lambda_{i1}\underbrace{[(1-t)v_1(0) + t\,v_1(1)]}_{v_1(t)} + \cdots + \lambda_{ik}\underbrace{[(1-t)v_k(0) + t\,v_k(1)]}_{v_k(t)} \tag{4.6}$$

$$v_i(t) = (1-t)\underbrace{[\lambda_{i1}v_1(0) + \cdots + \lambda_{ik}v_k(0)]}_{v_i(0)} + t\,\underbrace{[\lambda_{i1}v_1(1) + \cdots + \lambda_{ik}v_k(1)]}_{v_i(1)} \tag{4.7}$$

It remains to prove that M is planar. Observe that, since Γ_1 and Γ_2 are planar, no two vertices lie on the same point and hence no two vertices have the same convex coordinates. Also, let p be any point on any edge of G in Γ_1 (Γ_2), since the drawing is planar, then the convex coordinates of p are distinct from the convex coordinates of any internal vertex of G in Γ_1 (Γ_2).

Suppose for a contradiction that M is not planar. Then, there exists a time instant t during M such that:

(i) either two vertices v and w overlap, or

(ii) some vertex v lies on a point p traversed by an edge of G.

It follows that, the convex coordinates of vertex v must be equal either to the convex coordinates of vertex w or to those of point p, contradicting the hypothesis that the convex coordinates of any internal vertex of G remain fixed during the whole morph. □

Observe that this technique applies also in the degenerate case in which G has only two external vertices. Namely, graph G is a path $v_1, \ldots, v_k$, the external vertices are v_1 and v_k and the internal vertices are the intermediate vertices, whose position is expressed as a convex combination of those of v_1 and v_k.

Also, if P is a triangle, the convex coordinates of an internal point p of P are called *barycentric coordinates* and can be computed as follows.

Let a, b, and c be the three vertices bounding P and let p be an internal point of P. Denote by $\mathcal{S}_{abc}$ the area of the triangle bounded by a, b, and c. Analogously, denote by $\mathcal{S}_{pab}$, $\mathcal{S}_{pbc}$, and $\mathcal{S}_{pac}$ the areas of the triangles bounded by p and two consecutive vertices along the boundary of P (see Figure 4.4). Then the barycentric coordinates

α, β, and γ of p in P, associated with a, b, and c are computed as follows.

$$\alpha = \frac{\mathcal{S}_{pbc}}{\mathcal{S}_{abc}}; \qquad \beta = \frac{\mathcal{S}_{pac}}{\mathcal{S}_{abc}}; \qquad \gamma = \frac{\mathcal{S}_{pab}}{\mathcal{S}_{abc}}$$

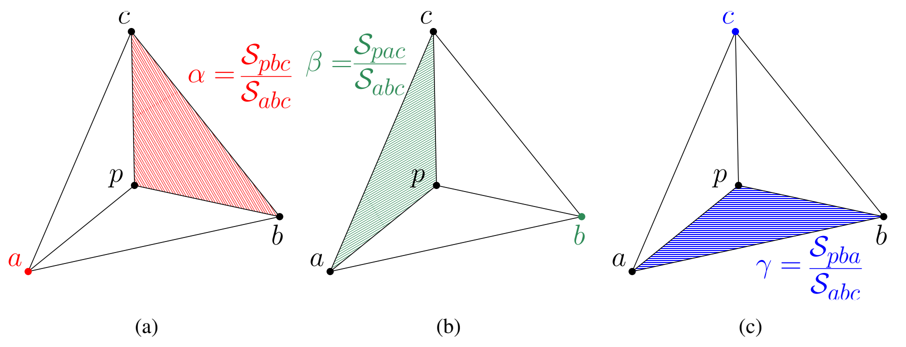

c
$\alpha = \frac{\mathcal{S}_{pbc}}{\mathcal{S}_{abc}}$
p
a
b
(a)
c
$\beta = \frac{\mathcal{S}_{pac}}{\mathcal{S}_{abc}}$
p
a
b
(b)
c
p
a
b
$\gamma = \frac{\mathcal{S}_{pba}}{\mathcal{S}_{abc}}$
(c)

Figure 4.4: Computing the barycentric coordinates of a point p inside a triangle.

4.5 Simulating the Rotation of a Triangle

In the following we show how the rotation of a triangle can be simulated in a constant number of steps. Let Δ_1 and Δ_2 be two planar straight-line drawings of the cycle (x, y, z) T. In the following we show that there exists a planar linear morph that transforms Δ_1 into Δ_2 in a constant number of steps. Assume that both Δ_1 and Δ_2 are inscribed in the same circumference C. Otherwise a translation and a scaling would suffice to achieve. Denote by x_1, y_1, and z_1, and by x_2, y_2, and z_2, the positions of vertices x, y, and z in Δ_1 and Δ_2, respectively. Also, for $i = 1, 2$, denote by $\widehat{x_i y_i}$, $\widehat{x_i z_i}$, and $\widehat{z_i y_i}$ the open arcs of C bounded by such vertices and not containing z_i, y_i, and x_i, respectively.

Observe that, the unique configuration of the two drawings in which none of the vertices can be directly moved to its final position is shown in Figure 4.5(a). Namely, x_2 lies along $\widehat{z_1 y_1}$, y_2 lies along $\widehat{x_1 z_1}$, and z_2 lies along $\widehat{z_1 y_1}$. As long as at least one of the vertices of T has visibility on its final destination, move it.

Otherwise, it suffice to simultaneously move x, y, and z along their incident edges, thus simulating a rotation. Namely, with a unique planar linear morphing step, we move x along $\overline{x_1 z_1}$, y along $\overline{y_1 x_1}$, and z along $\overline{z_1 y_1}$. Now, each vertex has direct

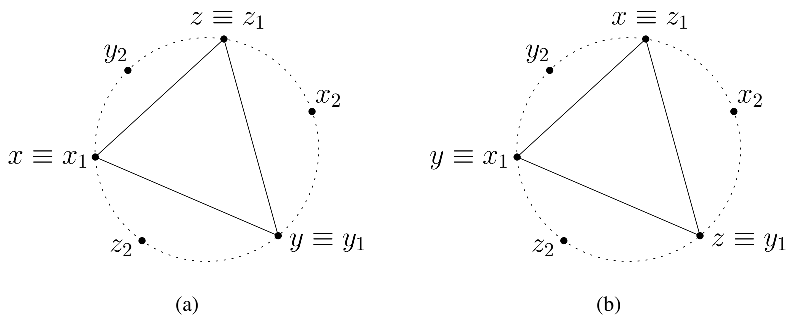


Figure 4.5: Morphing Δ_1 into Δ_2. (a) None of the vertices can be moved to its final position. (b) Simultaneously moving each vertex to the position occupied by its right neighbor allows for moving all vertices to their final position with a unique planar linear morphing step.

visibility on its final position and hence with a unique further planar linear morphing step the current drawing can be morphed into the final one.

4.6 Obtaining a Morph from a Pseudo-Morph

In this section[2] we show how a pseudo-morph $\mathcal{M}$ of a plane graph G can be extended to a morph M of G. Recall that a pseudo-morph $\langle \Gamma_s, \Gamma'_s = \Gamma'_1, \ldots, \Gamma'_q = \Gamma'_t, \Gamma_t \rangle$ of a plane graph G is defined as:

- the contraction of an x-contractible vertex v with $\deg(v) \leq 5$ onto a neighbor x in Γ_s;
- a morph $M' = \langle \Gamma'_s = \Gamma'_1, \ldots, \Gamma'_q = \Gamma'_t \rangle$ transforming the obtained drawing $\Gamma'_s = \Gamma'_1$ of the reduced graph $G' = G/(v,x)$ into a drawing $\Gamma'_q = \Gamma'_t$ of G'; and
- the uncontraction of v from x in $\Gamma'_q = \Gamma'_t$, resulting in Γ_t.

In the following we describe how $\mathcal{M}$ can be converted into an actual morph by uncontracting vertex v to a suitable position in each drawing Γ'_i of G', thus obtainig a straight-line planar drawing Γ_i of G.

[2]The contents of this section are joint work with Patrizio Angelini and Fidel Barrera-Cruz, and are an extension of [BHL13]

Denote by M_v the planar unidirectional morph $\langle \Gamma_1, \dots, \Gamma_q \rangle$ obtained by adding v to each drawing Γ_i of M'. The final morph M is obtained by concatenating:

- a planar linear morphing step moving v from its position in Γ_s to that in Γ_1;
- morph M_v; and
- a planar linear morphing step moving v from its position in Γ_q to that in Γ_t.

This strategy of constructing M starting from $\mathcal{M}$ by suitably placing v in each drawing of M' is the same that was applied in [AAC$^+$13, AFPR13, BHL13]. It should be noted that the algorithm for placing v in $\Gamma'_1, \dots, \Gamma'_q$ differs slightly in those three papers. We opt here for an extension of the one in [BHL13], as it ensures that M is a unidirectional morph with the same number of steps as $\mathcal{M}$, that is, $O(n)$. However, since in [BHL13] G is assumed to be maximal plane, vertex v can always be chosen to be an internal vertex of G with degree at least 3.

In order to guarantee the planarity of M_v (and hence of M), when adding back v to any drawing Γ'_i of M', vertex v must lie inside its kernel in Γ_i, that is, the uncontraction kernel of v in Γ'_i.

The algorithm described in [BHL13] finds a suitable point for v in each Γ'_i in a disk sector S_i centered at vertex x and lying in the interior of the polygon induced, in Γ'_i, by the neighbors of v, that is, delimited by two consecutive edges incident to x, or by their elongations emanating from x. Since, by Lemma 4.1, x lies on the boundary of the uncontraction kernel of v, the intersection between S_i and the uncontraction kernel of v is non-empty. Barrera-Cruz *et al.* [BHL13] prove that each sector S_i contains at least one *nice* point, defined as follows. All the points of S_q are nice. For $i = 1, \dots, q-1$, a point p_i of S_i is nice if there exists a nice point p_{i+1} in S_{i+1} such that the line passing through p_i and p_{i+1} is parallel to the trajectory followed by each vertex during the unidirectional morphing step transforming Γ'_i into Γ'_{i+1}. The proof in [BHL13] is completed by showing that placing v on the nice point p_i in Γ'_i and on the corresponding nice point p_{i+1} in Γ'_{i+1} yields two drawings Γ_i and Γ_{i+1} of G such that $\langle \Gamma_i, \Gamma_{i+1} \rangle$ is planar and, by construction, unidirectional.

We extend the algorithm of Barrera-Cruz *et al.* [BHL13] in order to handle the cases in which v is incident to the external face of G or $\deg(v) \in \{1, 2\}$.

At any time instant t during M', there exists an $\epsilon_t > 0$ such that the disk D_t centered at x with radius ϵ_t entirely lies inside the kernel of x [Fár48]. As a consequence, D_t does not contain any vertex of G other than x. Let ϵ be the minimum ϵ_t over all time instants t during M'. Denote by D_i the disk with radius ϵ centered at x in drawing Γ'_i, $1 \leq i \leq q$. Recall that, by construction, vertex x in G' is adjacent to all the neighbors of v in G and hence, by Lemma 4.1, it lies on the boundary of the uncontraction kernel of v in any straight-line planar drawing of G'.

In order to cover the cases in which $\deg(v) \in \{1, 2\}$, we give a definition of sectors S_i that is slightly different from the one given in [BHL13]. Namely, in each drawing Γ'_i of M', sector S_i is defined as the intersection between disk D_i and the uncontraction kernel of v. Recall that, by Lemma 4.1, vertex x lies on the boudnary of the uncontraction kernel of v in any straight-line planar drawing of G'. In the following lemma we prove that each sector S_i contains a non-empty set of nice points.

Lemma 4.10 *Let $M' = \langle \Gamma'_1, \dots, \Gamma'_q \rangle$ be a planar unidirectional morph of graph $G' = G/(v, x)$. For each $i = 1, \dots, q$, let D_i be a disk centered at vertex x in Γ'_i such that D_i completely lies in the kernel of v in Γ'_i and let S_i be the intersection between D_i and the uncontraction kernel of v in Γ'_i. Then, each sector S_i contains a non-empty set of nice points.*

Proof: If v and its neighbors induce in G an embedded wheel graph W such that v does not lie on the outer face of W, then S_i coincides with the disk sector defined in [BHL13]. Hence, it contains a non-empty set of nice points.

Denote by $\deg'(x)$ the degree of vertex x in G'. Assume that $\deg'(x) \geq 1$, as otherwise G would consist of a single edge and there exists a planar unidirectional morph between any pair of its drawings with a constant number of steps. We distinguish two cases.

Case 1: $\deg'(x) \geq 2$. Let y and z be the two neighbors of v such that y and x, and x and z are consecutive neighbors of v in G. Observe that, as in [BHL13], in any straight-line planar drawing Γ' of G', the two radii of D_i delimiting the boundary of the uncontraction kernel of v completely lie either on the segments representing edges (x, y) and (x, z) or on their elongations emanating from x, depending on the position of y and z with respect to x (see Figure 4.6). Hence, as in [BHL13], for each drawing Γ'_i of M', sector S_i is defined as the sector of D_i delimited by edge (x, y) and (x, z) (or their elongations) and having a non-empty intersection with the uncontraction kernel of v. Since the construction of S_i is the same as [BHL13], each sector S_i contains at least a nice point at which v can be placed in order to obtain Γ_i.

Case 2: $\deg'(x) = 1$. Let y be the unique neighbor of x in G'. In order to simplify the description, we distinguish two subcases.

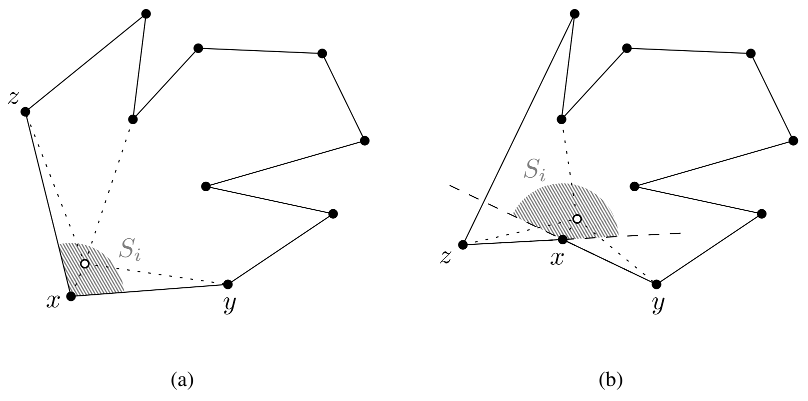


Figure 4.6: Illustration for Case 1. Sector S_i is delimited either by edges (x, y) and (x, z) (a), or by elongations (b).

Case 2a: $\mathbf{deg}(v) = 1$. Let y be the unique neighbor of x in G'. In the following we show how, in each drawing of M', vertex v can be placed and maintained at a point p_i of D_i lying on the elongation of edge (x, y) emanating from x.

Let d_i be the direction along the vertices of G' move during the planar linear morphing step transforming Γ'_i into Γ'_{i+1} and denote by y_i and y_{i+1} the points where vertex y lies in Γ'_i and Γ'_{i+1}, respectively. Let also p_i be a point on the elongation of edge (x, y) emanating from x in drawing Γ'_i and let p_{i+1} be the point at which the straight-line parallel to d_i passing through p_i and the straight-line passing through x and y_{i+1} intersect.

Since point p_i can always be chosen in S_i arbitrarily close to x on the elongation of edge (x, y) emanating from x in such a way that point p_{i+1} lies inside S_{i+1}, sector S_i contains at least a nice point.

Case 2b: $\mathbf{deg}(v) = 2$. Observe that, since in G' vertex x has an edge to any neighbor of v, in G vertex v is adjacent to vertices x and y only. We prove that S_i contains at least a nice point by reducing this case to cases $2a$ and 1.

Let G_z be the planar graph obtained by adding a new vertex z a new edge (x, z) to G'. Observe that $G' = G_z/(z, x)$. By applying the technique described in the previous case, we can augment M' to a planar unidirec-

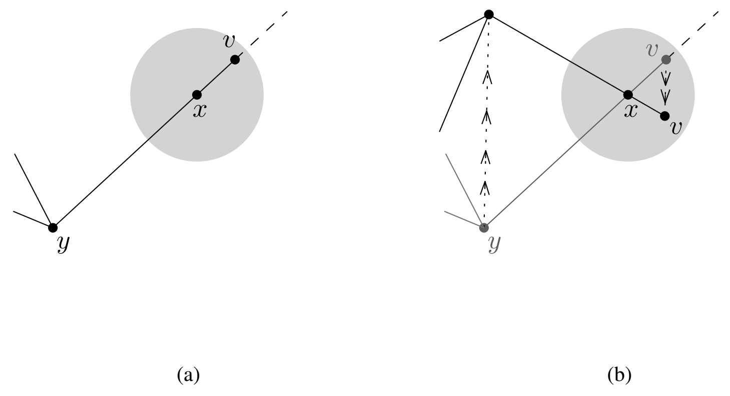


Figure 4.7: Vertex v is placed (a) and maintained (b) inside D_i, on the elongation of edge (x, y) at each step.

tional morph $M_z = \langle \Gamma_1^z, \dots, \Gamma_q^z \rangle$ of G_z such that z lies on the elongation of (x, z) in each drawing of M_z.

Let G_{vz} be the planar graph, obtained by adding vertex v and edges (v, x) and (v, y) to G_z, and let G'_{vz} be the graph obtained by contracting v onto x in G_{vz} and observe that in G'_{vz} vertex x has degree 2. Hence, by Case 1, we can find a nice point for v in each drawing of M_z. Finally, observe that for each $i = 1, \dots, q$ the uncontraction kernel of v is the same in Γ'_i and in Γ_i^z, thus proving the existence of a nice point for v in each S_i.

□

In order to prove that placing v at a nice point of S_i in each drawing Γ'_i of M' yields a planar unidirectional morph M_v, we just need to prove that vertex v does not introduce any crossings dueing M_v. Recall that, since v lies in S_i, it lies between the two segments r and p incident to x and delimiting the uncontraction kernel of v. Also, (i) the extrema of such segments move, at each step of M', at constant speed along straight-line parallel trajectories, and (ii) the straight-line l parallel to d_i and passing though v in Γ_i intersects r and p at points a and b, respectively.

Then, we just need to show that vertex v lies on the same side of r and p at any time during M_v. This can be done by applying Lemma 4 of [BHL13], which we restate as follows.

Lemma 4.11 (Lemma 4 of [BHL13]) *Let L be a horizontal line, and let x_0, x_1, y_0, and y_1 be points on L. Consider a point x that moves at constant speed from x_0 to x_1 in one unit of time. If y_i is to the right of x_i,$i = 0, 1$, and y is a point that moves at constant speed from y_0 to y_1 in one unit of time then y remains to the right of x during their movements. Note that x_0 may lie to the right or to the left of x_1 and the same holds for y_0 and y_1.*

Chapter 5

Morphing Series-Parallel Graphs

In this chapter[1] we show that, for any two planar straight-line drawings Γ_a and Γ_b of a plane series-parallel graph G, a linear number of steps, in the size of G, always suffice to morph Γ_a into Γ_b, as stated in the following theorem.

Theorem 5.1 *Let Γ_a and Γ_b be two planar straight-line drawings of an n-vertex plane series-parallel graph G. There exists a morph $M_{ab} = \langle \Gamma_a, \dots, \Gamma_b \rangle$ with $O(n)$ steps transforming Γ_a into Γ_b.*

In order to prove the theorem, we first construct a pseudo-morph $\mathcal{M}_{ab}$ with $O(n)$ steps by concatenating a pseudo-morph $\mathcal{M}_a$ from Γ_a to a *canonical drawing* Γ^* of G, and a pseudo-morph $\overline{\mathcal{M}_b}$, transforming Γ^* into Γ_b. Observe that $\overline{\mathcal{M}_b}$ is easily obtained by reverting the pseudo-morph $\mathcal{M}_b$ that transforms Γ_b into Γ^*. Finally, by recursively applying the technique described in Section 4.6, we convert pseudo-morph $\mathcal{M}_{ab} = \langle \Gamma_a, \dots, \Gamma^*, \dots, \Gamma_b \rangle$ into an actual morph $M_{ab} = \langle \Gamma_a, \dots, \Gamma_b \rangle$.

In the remainder of this chapter, we assume that the input graph G is biconnected. Observe that this is not a loss of generality, as Γ_a and Γ_b can be augmented to two drawings of the same plane biconnected series-parallel graph by repeatedly applying the technique described in Section 4.3 to add the same vertex and the same edges to both drawings.

In Section 5.1 we describe canonical drawings of a plane biconnected series-parallel graphs, while in Section 5.2 we show an algorithm to compute a pseudo-morph of a planar straight-line drawing Γ of a biconnected series-parallel graph G into its canonical drawing Γ^* with a linear number of steps.

[1]Part of the contents of this chapter are joint work with Patrizio Angelini, Fabrizio Frati, and Maurizio Patrignani, appeared in [AFPR13], and have been submitted to journal.

5.1 Canonical Drawings of Series-Parallel Graphs

Let e be an edge of G that is incident to the external face and denote by $T_e(G)$ the *SPQ-tree*, rooted at edge e (called *reference edge*), of G. Observe that, since series-parallel graphs do not have triconnected subgraphs, the SPQR-tree of series-parallel graphs is also called SPQ-tree, as it does not contain R-nodes. In the following we describe how to compute the canonical drawing Γ^* of G with respect to its reference edge e.

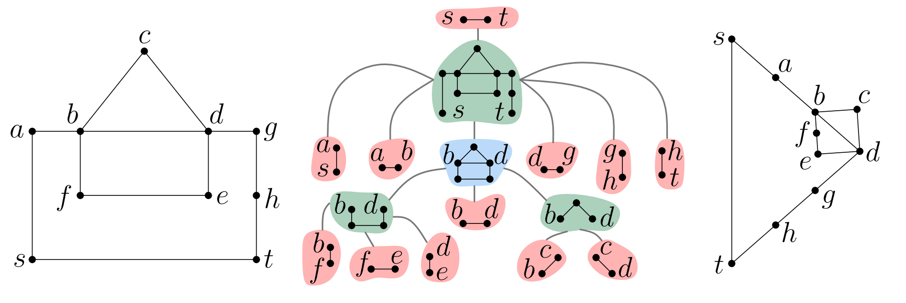


Figure 5.1: (a) A planar drawing of a plane series-parallel graph G. (b) The SPQ-tree $T_{(s,t)}$, rooted at edge (s, t), of G. Q-nodes, S-nodes, and P-nodes, are drawn as red, green, and blue regions, respectively. Inside the region representing each node μ is drawn $skel(\mu)$. (c) The canonical drawing Γ^*_G of G with respect to $T_{(s,t)}$.

Tree $T_e(G)$ is traversed top-down and, by exploiting the technique described in Section 2.2.1, a suitable geometric region of the plane (i.e., a diamond or a boomerang) is assigned to each node μ of T_e; such a region will contain the drawing of pertinent graph $pert(\mu)$ of μ. We assign boomerangs (either left or right, depending on the embedding of G) to S-nodes, and diamonds to P- and Q-nodes, as follows.

First, consider the Q-node ρ corresponding to the root edge e of G. Draw edge e as a segment between points $(0, 1)$ and $(0, -1)$. Also, if ρ is adjacent to an S-node μ in $T_e(G)$, then assign to μ the left boomerang $N = (0, 1), E = (-1, 0), S = (0, -1), W = (-2, 0)$ or the right boomerang $N = (0, 1), E = (2, 0), S = (0, -1), W = (1, 0)$, depending on the embedding of G; if ρ is adjacent to a P-node μ, then assign to μ the diamond $N = (0, 1), E = (+2, 0), S = (0, -1), W = (-2, 0)$. s Then, consider each node μ of $T_e(G)$ according to a top-down traversal.

If μ is an S-node (see Figure 5.2(a)), let N, E, S, W be the boomerang assigned to

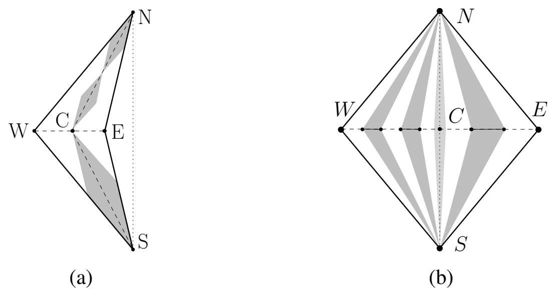


Figure 5.2: (a) Diamonds inside a boomerang. (a) Boomerangs (and a diamond) inside a diamond.

it and let α be the angle $\widehat{WNE}$. We assign diamonds to the children $\mu_1, \mu_2, \ldots, \mu_k$ of μ as follows. Consider the midpoint C of segment $\overline{WE}$. Subdivide $\overline{NC}$ into $\lceil \frac{k}{2} \rceil$ segments with the same length and $\overline{CS}$ into $\lfloor \frac{k}{2} \rfloor$ segments with the same length. For $i = 1, \ldots, k$, enclose segment $\overline{N_iS_i}$ into a diamond N_i, E_i, S_i, W_i, with $\widehat{W_iN_iE_i} = \alpha$, and assign this diamond to the child μ_i of μ.

If μ is a P-node (see Figure 5.2(b)), let N, E, S, W be the diamond assigned to it. Assign boomerangs and diamonds to the children $\mu_1, \mu_2, \ldots, \mu_k$ of μ as follows. If a child μ_l of μ is a Q-node, then left boomerangs are assigned to $\mu_1, \ldots, \mu_{l-1}$, right boomerangs are assigned to $\mu_{l+1}, \ldots, \mu_k$, and a diamond is assigned to μ_l. Otherwise, right boomerangs are assigned to all of $\mu_1, \mu_2, \ldots, \mu_k$. We assume that a child μ_l of μ that is a Q-node exists, the description for the case in which no child of μ is a Q-node being similar and simpler. We describe how to assign left boomerangs to the children $\mu_1, \mu_2, \ldots, \mu_{l-1}$ of μ. Consider the midpoint C of segment $\overline{WE}$ and consider $2l$ equidistant points $W = p_1, \ldots, p_{2l} = C$ on segment $\overline{WC}$. For $i = 1, \ldots, l-1$, assign the quadrilateral $N_i = N, W_i = p_{2i}, S_i = S, E_i = p_{2i+1}$ to the child μ_i of μ. Also, assign right boomerangs to $\mu_{l+1}, \mu_{l+2}, \ldots, \mu_k$ in a symmetric way. Finally, assign to μ_l any diamond such that $N_l = N, S_l = S$, W_l is any point between C and E_{l-1}, and E_l is any point between C and W_{l+1}.

If μ is a Q-node, let N, E, S, W be the diamond assigned to it. Draw the edge corresponding to μ as a straight-line segment between N and S.

We now argue that no two edges of G intersect in the canonical drawing Γ^*_G of G. Namely, let e_1 and e_2 be any two edges of G. Consider the lowest common ancestor ν of the Q-nodes τ_1 and τ_2 of $T_e(G)$ representing e_1 and e_2, respectively. Also, consider the children ν_1 and ν_2 of ν such that the subtree of T_e rooted at ν_i contains

τ_i, for $i = 1, 2$. Note that nodes ν_1 and ν_2 have been assigned internally-disjoint regions of the plane. Since the subgraphs G_1 and G_2 of G corresponding to ν_1 and ν_2, respectively, are entirely drawn inside such regions, it follows that e_1 and e_2 do not intersect except, possibly, at common endpoints.

5.2 Pseudo-Morphing to Canonical Drawing

In order to construct a pseudo-morph of a straight-line planar drawing Γ_G of G into its canonical drawing Γ^*_G, we do the following:

(i) We perform a contraction of a vertex v of G onto a neighbor of v, hence obtaining a drawing $\Gamma_{G'}$ of a graph G' with $n-1$ vertices;

(ii) we inductively construct a pseudo-morph from $\Gamma_{G'}$ to the canonical drawing $\Gamma^*_{G'}$ of G'; and

(iii) we uncontract v and perform a sequence of morphing steps to transform $\Gamma^*_{G'}$ into the canonical drawing Γ^*_G of G.

In the following we describe the three steps in more detail.

5.2.1 Step 1: Contract a vertex v

Let $T_e(G)$ be the decomposition tree of G rooted at some edge e incident to the external face of G. Assume that $T_e(G)$ contains at least one P-node (the case in which $T_e(G)$ does not contain any P-node is much simpler, hence it will be discussed later). Consider a P-node ν such that the subtree of $T_e(G)$ rooted at ν does not contain any other P-node. This implies that all the children of ν, with the exception of at most one Q-node, are S-nodes whose children are Q-nodes. Hence, the pertinent graph $pert(\nu)$ of ν is a series-parallel graph composed of a set of paths connecting its poles s and t, where at most one of these paths might be composed of a single edge (s, t).

Let p_1 and p_2 be two paths joining s and t whose union is a cycle $\mathcal{C}$ not containing other vertices in its interior (see Figure 5.3(a)). Such paths exist because the “rest of the graph” with respect to ν (namely, the graph induced by s, t, and the vertices of G that do not belong to $pert(\nu)$) lies in the external face of $pert(\nu)$, given that the root e of $T_e(G)$ is incident to the external face of G. Internally triangulate $\mathcal{C}$ by inserting dummy edges in its interior (dashed edges of Figure 5.3). Cycle $\mathcal{C}$ and the added dummy edges yield a drawing of a maximal outerplane graph O, which has at least two vertices of degree 2.

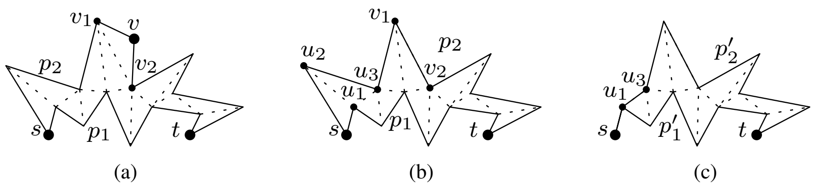


Figure 5.3: The internally triangulated cycle $\mathcal{C}$ formed by paths p_1 and p_2. Dummy edges are drawn as dashed lines. (a–b) Vertex v of degree 2 can be contracted onto v_1. (b–c) Vertex u_2 of degree 3 can be contracted onto u_1.

By Lemma 4.4, graph O contains a degree-2 vertex v that can be contracted onto any of its neighbors. We distinguish two cases, based on whether v is different from s and from t.

Case 1: *O* contains a degree-2 vertex v different from s and from t.
Assume, without loss of generality, that v belongs to p_2. Since O is internally triangulated, both the neighbors v_1 and v_2 of v belong to p_2, and they are joined by a dummy edge. We obtain Γ'_G from Γ_G by contracting v onto one of its neighbors, while preserving planarity (see Figures 5.3(a) and 5.3(b)). We further distinguish three subcases. Either p_2 contains more than two edges (*Case 1.1*) or p_2 consists of exactly two edges, namely (v_1, v) and (v, v_2). If the latter case holds, either edge (v_1, v_2) exists in G (*Case 1.2*) or not (*Case 1.3*). In the three cases we do the following.

***Case 1.1*:** Path p_2 is replaced in G' with a path p'_2 that contains edge (v_1, v_2) and does not contain vertex v.

***Case 1.2*:** Graph G' is set as $G \setminus \{v\}$.

***Case 1.3*:** Path p_2 is replaced in G' with edge (v_1, v_2).

Case 2: The only two vertices of degree 2 in *O* are s and t.
In this case, by Lemma 4.4, one of the two vertices u_1 and u_2 adjacent to s, say u_2, has degree 3 and is u_1-contractible (see Figures 5.3(b) and 5.3(c)). Let u_3 be the neighbor of u_1 and u_2 different from s. Let also p'_2 be the path composed of edge (u_1, u_3) and of the subpath of p_2 between u_3 and t, and let p'_1 be the subpath of p_1 between u_1 and t. Observe that G' contains edge (u_1, u_3) and does not contain vertex u_2. We obtain Γ'_G from Γ_G by contracting u_2 onto u_1.

Tree $T_e(G')$ is obtained from $T_e(G)$ by performing the local changes described hereunder, with respect to the above cases.

Case 1. Let τ_1 and τ_2 be the nodes of $T_e(G)$ corresponding to paths p_1 and p_2. Note that τ_2 is an S-node, as $v \in p_2$ and $v \neq s, t$. The two Q-nodes that are children of τ_2 and that correspond to edges (v, v_1) and (v, v_2) are removed from $T_e(G')$.

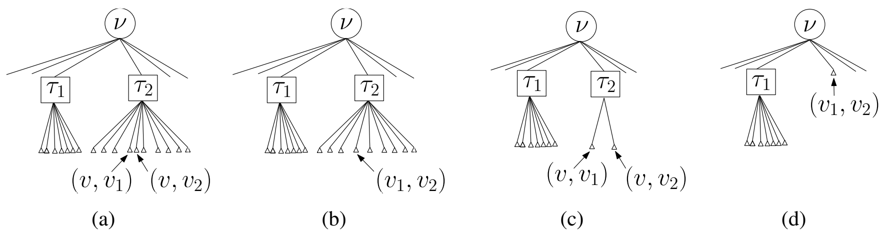


Figure 5.4: Construction of $T_e(G')$ starting from $T_e(G)$ in *Case 1.* (a–b) $T_e(G)$ and $T_e(G')$, respectively, in *Case 1.1.* (c–d) $T_e(G)$ and $T_e(G')$, respectively, in *Case 1.3.*

***Case 1.1*:** A Q-node corresponding to (v_1, v_2) is added to $T_e(G')$ as a child of τ_2 (see Figures 5.4(a) and 5.4(b)).

***Case 1.2*:** τ_2 is removed from $T_e(G')$. Also, if ν has no children other than τ_1 and τ_2 in $T_e(G')$, then ν is replaced with τ_1 in $T_e(G')$.

***Case 1.3*:** τ_2 is replaced in $T_e(G')$ with a Q-node corresponding to edge (v_1, v_2) (see Figures 5.4(c) and 5.4(d)).

Case 2. Let τ_1 and τ_2 be the nodes of $T_e(G)$ corresponding to paths p_1 and p_2, and let μ be the parent of ν. Note that τ_1 and τ_2 are S-nodes, as $u_1, u_2 \neq s, t$. First, the Q-nodes corresponding to edges (s, u_2) and (u_2, u_3) are removed from the children of τ_2, and a Q-node ν_Q (corresponding to edge (u_1, u_3)) is added to $T_e(G')$. We distinguish the cases in which ν has more than two children in $T_e(G)$ (*Case 2.1*) and when ν has exactly two children in $T_e(G)$ (*Case 2.2*).

***Case 2.1*:** An S-node ν_S and a P-node ν_P are introduced in $T_e(G')$, in such a way that (i) ν_S is a child of ν, (ii) the Q-node corresponding to (s, u_1) and ν_P are children of ν_S, (iii) τ_1 and τ_2 are children of ν_P, and (iv) ν_Q is a child of τ_2 (see Figures 5.5(a) and 5.5(b)).

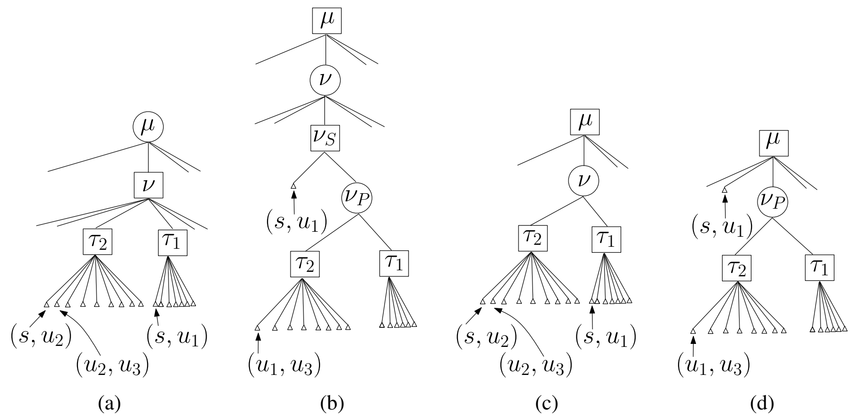


Figure 5.5: Construction of $T_e(G')$ starting from $T_e(G)$ in *Case 2*. (a–b) $T_e(G)$ and $T_e(G')$, respectively, in *Case 2.1*. (c–d) $T_e(G)$ and $T_e(G')$, respectively, in *Case 2.2*.

Case 2.2: Node ν is removed from the children of μ, and a P-node ν_P is introduced in $T_e(G')$ in such a way that (i) the Q-node corresponding to (s, u_1) and ν_P are children of μ, (ii) τ_1 and τ_2 are children of ν_P, and (iii) ν_Q is a child of τ_2 (see Figures 5.5(c) and 5.5(d)).

5.2.2 Step 2: Apply induction to construct the canonical drawing $\Gamma^*_{G'}$ of G'

Let Γ'_G be the drawing of the graph $G' = G/(v, x)$ obtained after the contraction of vertex v performed in *Case 1* or in *Case 2*.

Inductively construct a morph from $\Gamma_{G'}$ to the canonical drawing $\Gamma^*_{G'}$ of G' in $c \cdot (n-1)$ steps, where c is a constant.

Step 3: Uncontract vertex v and construct a canonical drawing of G

We describe how to obtain Γ^*_G from $\Gamma^*_{G'}$ by uncontracting v and performing a constant number of morphing steps.

Case 1: O contains a degree-2 vertex v different from s and t.

Case 1.1: Drawings $\Gamma^*_{G'}$ and Γ^*_G coincide except for the fact that path p_2 in Γ^*_G contains v, while path p'_2 in $\Gamma^*_{G'}$ does not contain v. Paths p'_2 and p_2

are drawn inside two equal boomerangs in $\Gamma^*_{G'}$ and in Γ^*_G, respectively; however, v and some of the vertices of p'_2 need to be moved in order to obtain the drawing of p_2 as in $\Gamma^*_{G'}$. Namely, the drawing $\Gamma^*_{p'_2}$ of p'_2 inside the boomerang N, E, S, W assigned to τ_2 in $\Gamma^*_{G'}$ is composed of edges lying on two straight-line segments $\overline{NC}$ and $\overline{SC}$, where C is the midpoint of segment $\overline{EW}$ (see Figure 5.6(a)). The drawing $\Gamma^*_{p_2}$ of p_2 in Γ^*_G also lies inside N, E, S, W and is composed of edges lying on $\overline{NC}$ and $\overline{SC}$, but vertices lie on different points (see Figure 5.6(e)).

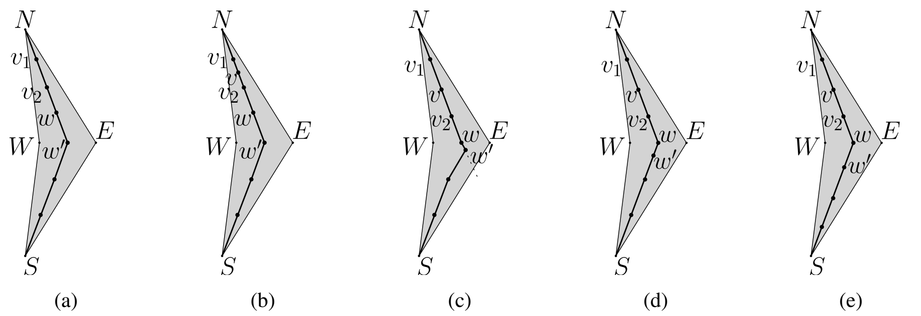


Figure 5.6: Construction of Γ^*_G from $\Gamma^*_{G'}$ when *Case* 1.1 applied. (a) $\Gamma^*_{p'_2}$. The boomerang assigned to τ_2 is light-grey. (b) Vertex v is uncontracted and placed on segment $\overline{v_1v_2}$. (c) Vertices on the path between s and w are placed in their final position, and vertex w' is placed arbitrariliy close to C on the elongation of $\overline{NC}$. (d) Vertex w' is placed on $\overline{CS}$. (e) Vertices on the path between w and t are placed in their final position, hence obtaining Γ^*_G.

With one morphing step, uncontract v from the vertex it had been contracted onto and place it on any point of segment $\overline{v_1v_2}$ (note that edge (v_1, v_2) exists in G' and not in G; see Figure 5.6(b)). Then, in order to redistribute the vertices of p_2 on $\overline{NC}$ and $\overline{SC}$, perform the following operation. Assume without loss of generality that s is on point N and t is on point S in $\Gamma^*_{G'}$ and in Γ^*_G. Consider the vertices $w \in p_2$ and $w' \in p'_2$ that are placed on point C in Γ^*_G and $\Gamma^*_{G'}$, respectively. Note that either $w = w'$ or $(w, w') \in p_2$. If $w = w'$, either the subpath $p_2(s, w)$ of p_2 between s and w or the subpath $p_2(w, t)$ of p_2 between w and t has the same drawing in Γ^*_G and $\Gamma^*_{G'}$, say $p_2(w, t)$ has such a property. With one morphing step move the vertices of $p_2(s, w)$ on segment $\overline{NC}$ till reaching their positions in Γ^*_G. If $w \neq w'$, assume without loss of generality that $w \in p_2(s, w')$. With one morphing step, move the vertices of $p_2(s, w)$ and vertex w' along the line through N and C, so that the vertices of $p_2(s, w)$ reach their positions in Γ^*_G and w' is placed arbitrarily close to C on the elongation of $\overline{NC}$ (see Figure 5.6(c)). With a second morphing step, move w' to any point of $\overline{SC}$ between w and its other neighbor in p_2 (see Figure 5.6(d)). Finally, with a third morphing step, move the vertices of $p_2(w, t)$ on segment $\overline{SC}$ till reaching their positions in Γ^*_G (see Figure 5.6(e)).

***Case 1.2* and *Case 1.3*:** Note that $\Gamma^*_{G'}$ and Γ^*_G coincide, except for the fact that: (i) Γ^*_G contains one boomerang more than $\Gamma^*_{G'}$ (the one assigned to τ_2) inside the diamond assigned to ν, (ii) Γ^*_G might not contain the diamond assigned to the Q-node corresponding to edge (s, t) (in *Case 1.3*), and (iii) the boomerangs inside the diamond assigned to ν have a different drawing in $\Gamma^*_{G'}$ and Γ^*_G. Drawing $\Gamma^*_{G'}$ is illustrated in Figure 5.7(a), drawing Γ^*_G in *Case 1.2* is illustrated in Figure 5.7(c), and drawing Γ^*_G in *Case 1.3* is illustrated in Figure 5.7(d). Since edge (v_1, v_2) exists in G', its drawing in $\Gamma^*_{G'}$ is the straight-line segment between the points N' and S' of a diamond N', E', S', W'. Also, the drawing $\Gamma^*_{p_2}$ of p_2 in Γ^*_G lies inside a boomerang N, E, S, W with $N = N'$ and $S = S'$.

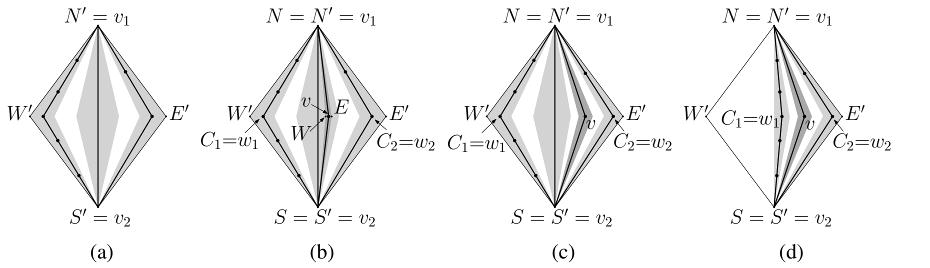


Figure 5.7: Construction of Γ^*_G from $\Gamma^*_{G'}$ when either *Case 1.2* or *Case 1.3* applied. (a) $\Gamma^*_{G'}$. The diamond assigned to edge (v_1, v_2) and the boomerangs assigned to the children τ_i of ν are light-grey. (b) Two points W and E are selected on $\overline{E'W'}$, creating a (dark gray) boomerang assigned to τ_2, and vertex v is moved to the midpoint of $\overline{EW}$. (c) Γ^*_G in *Case 1.2*, where edge (v_1, v_2) exists in G. (d) Γ^*_G in *Case 1.3*.

In order to transform $\Gamma^*_{G'}$ into Γ^*_G, initially place points W and E on segment $\overline{W'E'}$, on the same side with respect to segment $\overline{N'S'}$ (in *Case*

1.2, the side depends on the order of the children of ν in $T_e(G)$). With one morphing step, uncontract v from v_1 or v_2 and move it to the midpoint of segment $\overline{WE}$ (see Figure 5.7(b)). Observe that this morphing step does not cause crossings and preserves the embedding of G. Namely, the fact that v was contracted onto v_1 or v_2 and that edges (v, v_1) and (v, v_2) exist in G ensures that path (v_1, v, v_2) lie either in a face delimited by edge (v_1, v_2) (in *Case 1.2*) or in the face that contains the dummy edge (v_1, v_2) (in *Case 1.3*).

Consider the children τ_i of ν in $T_e(G)$ that are not Q-nodes, with $i = 1, \dots, q$, and note that the drawing of each τ_i is composed of two straight-line segments $\overline{NC_i}$ and $\overline{SC_i}$. With a second morphing step, move the vertex w_i of τ_i lying on C_i, for each $i = 1, \dots, q$, and vertex v along the line through $\overline{WE}$ till reaching their positions in Γ^*_G. In the same morphing step, for each $i = 1, \dots, q$, the vertices on the path between s and w_i are moved as linear combination of the movements of s and w_i, and the vertices on the path between t and w_i are moved as linear combination of the movements of t and w_i. Hence, at the end of the morphing step, also such vertices reach their positions in Γ^*_G (see Figures 5.7(c) and 5.7(d)).

Case 2: The only two vertices of degree 2 in O are s and t.

Case 2.1: Note that $\Gamma^*_{G'}$ and Γ^*_G coincide, except for the drawing of p_1, p_2, p'_1, and p'_2.

Namely, p_1 and p_2 are drawn in Γ^*_G in two boomerangs N_1, W_1, S_1, WE_1 and $N_2 = N_1, W_2, S_2 = S_1, E_2$ lying inside the diamond assigned to ν (see Figure 5.8(d)), while p'_1 and p'_2 are drawn in $\Gamma^*_{G'}$ in two boomerangs $N'_1, W'_1, S'_1 = S_1, E'_1$ and $N'_2 = N'_1, W'_2, S'_2 = S'_1 = S_1, E'_2$ lying inside a diamond assigned to ν_P, that lies inside a boomerang N_S, W_S, S_S, E_S assigned to ν_S (with $S_S = S'_2 = S'_1 = S_1$), that lies inside the diamond assigned to ν (see Figure 5.8(a)).

Note that, since ν_S has two children in $T_e(G')$, vertex u_1 is placed on the midpoint C_S of segment $\overline{W_S E_S}$, that is, $C_S = N'_1 = N'_2$.

Let w_1 and w_2 be the vertices of p'_1 and p'_2, respectively, placed on the midpoints C'_1 and C'_2 of segments $\overline{W'_1 E'_1}$ and $\overline{W'_2 E'_2}$.

With one morphing step, move w_1 to any point C^1_S of $\overline{W_S C_S}$, move w_2 to any point C^2_S of $\overline{C_S E_S}$, and move u_1 to any point of segment $\overline{N_S C^2_S}$ (see Figure 5.8(b)). In the same morphing step, for each two vertices $a, b \in \{w_1, w_2, u_1, t\}$, the vertices lying on segment $\overline{ab}$ are moved as

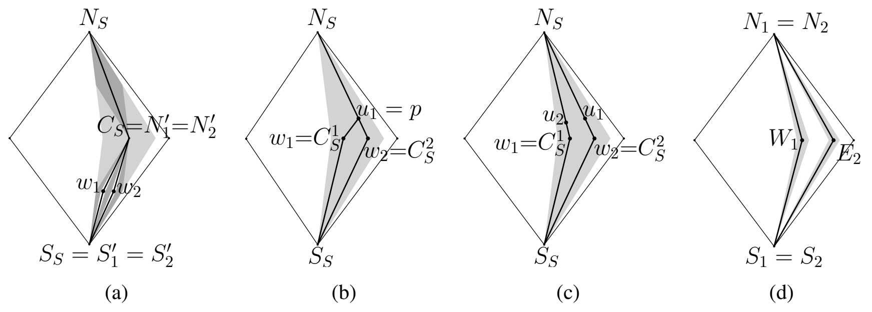


Figure 5.8: Construction of Γ^*_G from $\Gamma^*_{G'}$ when *Case* 2.1 applied. (a) $\Gamma^*_{G'}$. The boomerang assigned to ν_S is light-grey, the diamonds assigned to ν_P and to the Q-node corresponding to (s, u_2) are dark-grey, and the boomerangs assigned to τ_1 and τ_2 are white. (b) Vertices w_1 and w_2 are moved to points C_S^1 and C_S^2, and u_1 is moved to a point of $\overline{N_S C_S^2}$. (c) Vertex v is uncontracted from u_2 and moved to a point of $\overline{N_S C_S^1}$. (d) Γ^*_G. The boomerangs assigned to τ_1 and τ_2 are light-grey.

linear combination of the movements of a and b. Hence, at the end of the morphing step, all these vertices still lie on $\overline{ab}$.

Next, with one morphing step, uncontract u_2 from u_1 and move it to any internal point of segment $\overline{N_S C_S^1}$ (see Figure 5.8(c)). In the same morphing step, the vertices lying on segment $\overline{u_2 w_1}$ are moved as linear combination of the movements of u_2 and w_1.

Further, perform the same operation as in *Case 1.1* to redistribute the vertices of p_1 on $\overline{N_S C_S^1}$ and $\overline{S_S C_S^1}$, and the vertices of p_2 on $\overline{N_S C_S^2}$ and $\overline{S_S C_S^2}$. After this step, for each child τ_i of ν, the vertex w_i of τ_i lying on segment $\overline{W_S E_S}$ in Γ^*_G lies on $\overline{W_S E_S}$ also in the current drawing.

Finally, perform the same operation as in *Case 1.2* to move the vertex w_i of each child τ_i of ν to its final position (on segment $\overline{W_S E_S}$) in Γ^*_G. In the same morphing step, the vertices on the path between s and w_i are moved as linear combination of the movements of s and w_i, and the vertices on the path between t and w_i are moved as linear combination of the movements of t and w_i. Hence, at the end of the morphing step, also such vertices reach their positions in Γ^*_G.

***Case 2.2*:** Note that $\Gamma^*_{G'}$ and Γ^*_G coincide, except for the drawing of p_1, p_2, p'_1, and p'_2.

Namely, p_1 and p_2 are drawn in Γ^*_G inside two boomerangs (assigned to τ_1 and τ_2) lying inside the diamond assigned to ν (see Figure 5.9(b)), that lies inside the boomerang assigned to μ. Also, p'_1 and p'_2 are drawn in $\Gamma^*_{G'}$ inside two boomerangs lying inside a diamond (assigned to ν_P) that lies inside the boomerang assigned to μ (see Figure 5.9(a)). However, the boomerang assigned to μ in Γ^*_G has one diamond less than in $\Gamma^*_{G'}$, since in $\Gamma^*_{G'}$ it also contains the diamond assigned to edge (s, u_1). Also, the vertices in the boomerangs assigned to τ_1 and τ_2 have different positions in $\Gamma^*_{G'}$ and in Γ^*_G, since vertex u_2 is not present in $\Gamma^*_{G'}$.

With three morphing steps analogous to those performed in *Case 1.1*, redistribute the vertices inside the boomerang N, W, S, E assigned to μ in such a way that the vertex lying on the midpoint C of $\overline{WE}$ is the same in $\Gamma^*_{G'}$ and in Γ^*_G. Note that, after these steps, the diamonds assigned to ν_P and to edge (s, u_1) lie on the same segment, either $\overline{NC}$ or $\overline{SC}$, say $\overline{SC}$, and that the vertices lying on segment $\overline{NC}$ already are at their final position in Γ^*_G. Then, with three morphing steps analogous to those performed in *Case 2.1*, uncontract u_2 and collapse the two diamonds assigned to ν_P and to (s, u_1) into a single diamond. Then, with one morphing step (analogous to one of the steps performed in *Case* 1.1), move the vertices lying on segment $\overline{SC}$ till they reach their final position in Γ^*_G.

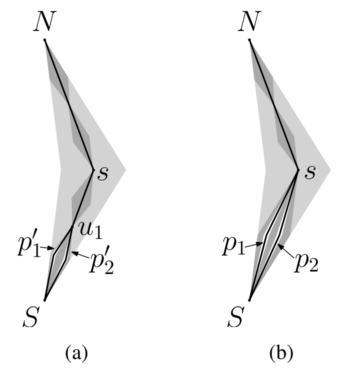


Figure 5.9: Construction of Γ^*_G from $\Gamma^*_{G'}$ in *Case* 2.2. (a) $\Gamma^*_{G'}$. The boomerang assigned to μ is light-grey, the diamonds assigned to the children of μ, including ν_P and the Q-node corresponding to (s, u_1) are dark-grey, and the boomerangs assigned to τ_1 and τ_2 are white. (b) Γ^*_G.

To conclude the description of the algorithm, it remains to consider the case in which $T_e(G)$ has no P-node. In such a case, we have that G is a simple cycle and that $T_e(G)$ contains exactly one S-node. If G has exactly three vertices, then a morph from any drawing Γ_G to the canonical drawing Γ^*_G can be easily computed in a constant number of steps by applying the technique described in Section 4.5. If G has more than three vertices, then we apply a procedure very similar to the one described in the case in which $T_e(G)$ contains a P-node. That is, we first triangulate the interior of G, thus obtaining a biconnected outerplane graph O; we then find a vertex v of degree 2 in O different from s and t, which always exists since s and t are adjacent, by construction. Then, we contract v on one of its neighbors, thus obtaining a drawing $\Gamma_{G'}$ of a graph G' with one vertex less than G; since $v \neq s, t$, graph G' is still a simple cycle. Next, we inductively construct a pseudo-morph from $\Gamma_{G'}$ to the canonical drawing $\Gamma^*_{G'}$ of G'; finally, we uncontract v and construct a morph between $\Gamma^*_{G'}$ and Γ^*_G. We remark that, since s and t are adjacent in G and in O, a vertex v of degree 2 different from s and t always exists in O.

5.2.3 Total Number of Steps

Consider the pseudo-morph $\mathcal{M} = \langle \Gamma_G, \dots, \Gamma^*_G \rangle$ transforming a planar straight-line drawing Γ_G of plane graph G into the canonical drawing Γ^*_G of G. Denote by M the actual morph $\langle \Gamma_G, \dots, \Gamma^*_G \rangle$ obtained from $\mathcal{M}$ by recursively applying the technique described in Section 4.6. Observe that the conversion of $\mathcal{M}$ into M does not introduce any further linear morphing step, as the contraction and the uncontraction of a vertex are replaced, respectively, with a single planar linear morphing step, while the motion of the contracted vertex during the morph of the resulting graph is computed according with the motion of the neighbor it has been contracted onto.

Let then v be the vertex of G contracted onto its neighbor u in the first phase of the algorithm and observe that graph $G' = G/(v, u)$ and its drawing $\Gamma_{G'}$ are obtained with a single contraction. In the second phase, a pseudo-morph $\mathcal{M}' = \langle \Gamma_{G'}, \dots, \Gamma^*_{G'} \rangle$ is computed with $T(n-1)$ planar linear morphing steps. In the third phase, drawing Γ^*_G is obtained from $\Gamma^*_{G'}$ with an uncontraction and a constant number of planar linear morphing steps. The total number $T(n)$ of planar linear morphing steps required by M is then $T(n) = T(n-1) + k$, where k is a constant. Since in the base case, namely when G' consists of a single edge, the morph to the canonical drawing is computed in a constant number of steps, it follows that $T(n) \in O(n)$.

Chapter 6

A Lower Bound

In this chapter[1] we show that, for every $n > 0$, there exist two drawings Γ_s and Γ_t of the same n-vertex plane graph, in fact a path, such that any planar morph between Γ_s and Γ_t consists of $\Omega(n)$ morphing steps. To the best of our knowledge, no super-constant lower bound was previously known.

The idea behind the lower bound is that linear morphs can poorly simulate rotations, that is, a morphing step rotates an edge of an angle whose size is $O(1)$. We then consider two drawings Γ_s and Γ_t of an n-vertex path P, where Γ_s lies on a straight-line, whereas Γ_t has a spiral-like shape, and we prove that in any planar morph between Γ_s and Γ_t there is one edge of P whose total rotation describes an angle whose size is $\Omega(n)$.

The lower bound described in this chapter implies that the algorithms described in Chapter 5 and in [Ros10] are asymptotically optimal.

The chapter is organized as follows. In Section 6.1 we describe the setting and give some preliminary definitions; in Section 6.2 we describe the natural constraints that any morph transforming Γ_s into Γ_t should satisfy; and in Section 6.3 we prove the lower bound.

6.1 Construction of the Drawings

In this section we show how to construct two straight-line planar drawings Γ_s and Γ_t of an n-vertex path $P = (v_1, \ldots, v_n)$, such that, as we will prove in Section 6.3, any planar morph M between Γ_s and Γ_t requires $\Omega(n)$ morphing steps. In order to

[1]The contents of this chapter are joint work with Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani.

simplify the description, we consider each edge $e_i = (v_i, v_{i+1})$ as oriented from v_i to v_{i+1}, for $i = 1, \dots, n-1$.

Drawing Γ_s (see Figure 6.1(a)) is such that all the vertices of P lie on a horizontal straight-line with v_i to the left of v_{i+1}, for each $i = 1, \dots, n-1$.

Drawing Γ_t (see Figure 6.1(b)) is such that:

- for each $i = 1, \dots, n-1$ with $i \mod 3 \equiv 1$, the (green) segment representing e_i is horizontal with v_i to the left of v_{i+1};
- for each $i = 1, \dots, n-1$ with $i \mod 3 \equiv 2$, the (blue) segment representing e_i is parallel to line $y = \tan(\frac{2\pi}{3})x$ with v_i to the right of v_{i+1}; and
- for each $i = 1, \dots, n-1$ with $i \mod 3 \equiv 0$, the (red) segment representing e_i is parallel to line $y = \tan(-\frac{2\pi}{3})x$ with v_i to the right of v_{i+1}.

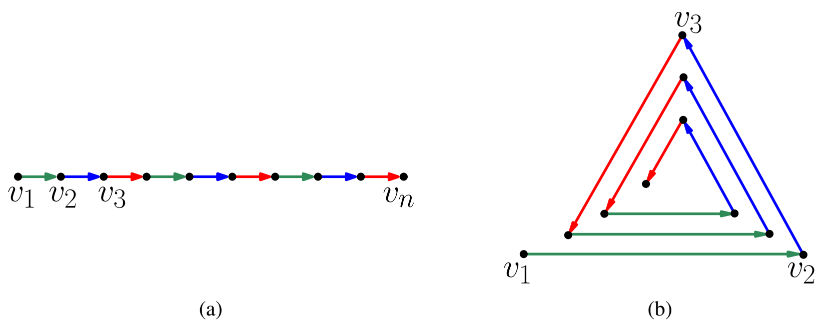


Figure 6.1: Drawings Γ_s (a) and Γ_t (b).

6.2 Definitions and Constraints

Let $M = \langle \Gamma_s = \Gamma_1, \dots, \Gamma_m = \Gamma_t \rangle$ be any planar morph transforming Γ_s into Γ_t.

For $i = 1, \dots, n$ and $j = 1, \dots, m$, we denote by v_i^j the point where vertex v_i is placed in Γ_j; also, for $i = 1, \dots, n-1$ and $j = 1, \dots, m$ we denote by e_i^j the directed straight-line segment representing edge e_i in Γ_j.

For $1 \leq j \leq m-1$, we define the *rotation* ρ_i^j of e_i around v_i during the morphing step $\langle \Gamma_j, \Gamma_{j+1} \rangle$ as follows (see Figure 6.2). Translate e_i at any time instant of $\langle \Gamma_j, \Gamma_{j+1} \rangle$ so that v_i stays fixed at a point a during the entire morphing step. After this translation, the morph between e_i^j and e_i^{j+1} is a rotation of e_i around a (where e_i might vary its length during $\langle \Gamma_j, \Gamma_{j+1} \rangle$) spanning an angle ρ_i^j, where we assume $\rho_i^j > 0$ if the rotation is counter-clockwise, and $\rho_i^j < 0$ if the rotation is clockwise.

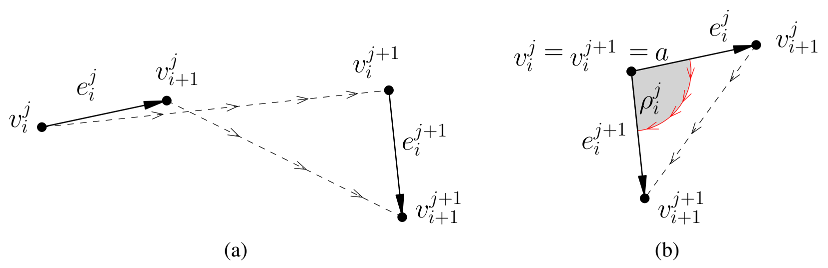


Figure 6.2: Rotation ρ_i^j. (a) Morph between e_i^j and e_i^{j+1}. (b) Translation of the positions of e_i during $\langle \Gamma_j, \Gamma_{j+1} \rangle$, resulting in e_i spanning an angle ρ_i^j around v_i.

With next lemma we prove that, during a single planar linear morphing step the rotation of an edge is, in modulus, strictly less than π.

Lemma 6.1 *For each $j = 1, \ldots, m-1$ and $i = 1, \ldots, n-1$, we have $|\rho_i^j| < \pi$.*

Proof: Suppose, for a contradiction, that, for some $1 \leq j \leq m-1$ and $1 \leq i \leq n-1$, the planar linear morphing step $\langle \Gamma_j, \Gamma_{j+1} \rangle$ of M is such that for edge e_i it holds $\rho_i^j \geq \pi$. Assume, without loss of generality, that:

- $v_i^j \equiv v_i^{j+1}$, that is, vertex v_i does not move during $\langle \Gamma_j, \Gamma_{j+1} \rangle$. Otherwise we can consider a drawing Γ'_{j+1} obtained by translating Γ_{j+1} in such a way that v_i has the same position both in Γ_j and in Γ'_{j+1}. Observe that morph $\langle \Gamma_j, \Gamma_{j+1} \rangle$ is a planar linear morph if and only if $\langle \Gamma_j, \Gamma'_{j+1} \rangle$ is. Also, the rotation of edge e_i is the same in both morphs.
- The morphing step $\langle \Gamma_j, \Gamma_{j+1} \rangle$ happens between time instants $t = 0$ and $t = 1$.

For any $0 \leq \bar{t} \leq 1$, denote by $v_i(\bar{t})$, $v_{i+1}(\bar{t})$, $e_i(\bar{t})$, and $\rho_i^j(\bar{t})$ the position of v_i, the position of v_{i+1}, the drawing of e_i, and the rotation of e_i around v_i at time instant t, respectively. Note that $v_i(0) = v_i^j = v_i^{j+1} = v_i(1)$, $v_{i+1}(0) = v_{i+1}^j$, $v_{i+1}(1) = v_{i+1}^{j+1}$, $e_i(0) = e_i^j$, $e_i(1) = e_i^{j+1}$, $\rho_i^j(0) = 0$, and $\rho_i^j(1) = \rho_i^j$. Also, denote by $\Gamma_{\bar{t}}$ the drawing of P at time $\bar{t}$ during $\langle \Gamma_j, \Gamma_{j+1} \rangle$.

Recall that, by hypothesis, $\langle \Gamma_j, \Gamma_{j+1} \rangle$ is a planar linear morphing step. We prove that there exists an intermediate drawing Γ_{t_r} of $\langle \Gamma_j, \Gamma_{j+1} \rangle$ in which vertices v_i and v_{i+1} overlap, thus contradicting the planarity of $\langle \Gamma_j, \Gamma_{j+1} \rangle$, and hence of M.

Since a morph is a continuous transformation, we have that for any $0 \leq \bar{t} \leq 1$, the planar linear morphing step $\langle \Gamma_j, \Gamma_{j+1} \rangle$ can be subdivided into two planar linear morphing steps $\langle \Gamma_i, \Gamma_{\bar{t}} \rangle$ and $\langle \Gamma_{\bar{t}}, \Gamma_{j+1} \rangle$. Observe that, since $\Gamma_{\bar{t}}$ is an intermediate drawing of $\langle \Gamma_j, \Gamma_{j+1} \rangle$, such a morphing step is planar if and only if both the linear morphs $\langle \Gamma_i, \Gamma_{\bar{t}} \rangle$ and $\langle \Gamma_{\bar{t}}, \Gamma_{j+1} \rangle$ are planar.

Also, since $|\rho_i^j| \geq \pi$, there exists a time instant t_π, with $0 < t_\pi \leq 1$, such that $|\rho_i^j(t_\pi)| = \pi$.

Consider drawing Γ_{t_π}. Since $|\rho_i^j(t_\pi)| = \pi$, it follows that $e_i(t_\pi)$ is parallel to $e_i(0)$ and oriented in the opposite way. Also, since by assumption vertex v_i does not move during the morph, v_i lies along the straight-line segment $\overline{v_{i+1}(0)v_{i+1}(t_\pi)}$, that is the trajectory of vertex v_{i+1} during morph $\langle \Gamma_i, \Gamma_{t_\pi} \rangle$. It follows that, during $\langle \Gamma_i, \Gamma_{t_\pi} \rangle$ there exists a time instant t_r, with $0 < t_r \leq t_\pi$, in which $v_i(t_r)$ and $v_{i+1}(t_r)$ coincide, thus contradicting the fact that $\langle \Gamma_i, \Gamma_{t_\pi} \rangle$ is planar, and hence the hypothesis that $\langle \Gamma_j, \Gamma_{j+1} \rangle$. □

For $j = 1, \ldots, m-1$, we denote by M_j the subsequence $\langle \Gamma_1, \ldots, \Gamma_{j+1} \rangle$ of M; also, for $i = 1, \ldots, n-1$, we define the *total rotation* $\rho_i(M_j)$ of edge e_i around v_i during morph M_j as $\rho_i(M_j) = \sum_{m=1}^{j} \rho_i^m$.

We will show in Lemma 6.3 that there exists an edge e_i, for some $1 \leq i \leq n-1$, whose total rotation $\rho_i(M_{m-1}) = \rho_i(M)$ is $\Omega(n)$. In order to do that, we first analyze the relationship between the total rotation of two consecutive edges of P.

Lemma 6.2 *For each $j = 1, \ldots, m-1$ and for each $i = 1, \ldots, n-2$, we have that $|\rho_{i+1}(M_j) - \rho_i(M_j)| < \pi$.*

Proof: Suppose, for a contradiction, that $|\rho_{i+1}(M_j) - \rho_i(M_j)| \geq \pi$ for some $1 \leq j \leq m-1$ and $1 \leq i \leq n-2$. Assume that j is minimal under this hypothesis. Since each vertex moves continuously during M_j, there exists an intermediate drawing Γ^* of P, occurring during morphing step $\langle \Gamma_j, \Gamma_{j+1} \rangle$, such that $|\rho_{i+1}(M^*) - \rho_i(M^*)| = \pi$, where $M^* = \langle \Gamma_1, \ldots, \Gamma_j, \Gamma^* \rangle$ is the morph obtained by concatenating M_{j-1} with the morphing step transforming Γ_j into Γ^*. Recall that in Γ_1 edges e_i and e_{i+1} lie on the same straight line and have the same orientation. Then, since $|\rho_{i+1}(M^*) - \rho_i(M^*)| = \pi$, in Γ^* edges e_i and e_{i+1} are parallel and have opposite orientations. Also, since edges e_i and e_{i+1} share vertex v_{i+1}, they lie on the same line. This implies that such edges overlap, contradicting the hypothesis that M^*, M_j, and M are planar. □

6.3 Proof of the Lower Bound

We are now ready to prove the key lemma for the lower bound.

Lemma 6.3 *There exists an index i such that $|\rho_i(M)| \in \Omega(n)$.*

Proof: Refer to Figure 6.1. For every $1 \leq i \leq n-2$, edges e_i and e_{i+1} form an angle of π radiants in Γ_s, while they form an angle of $\frac{\pi}{3}$ radiants in Γ_t. Hence, $\rho_{i+1}(M) = \rho_i(M) + \frac{2\pi}{3} + 2z_i\pi$, for some $z_i \in \mathbb{Z}$.

In order to prove the lemma, it suffices to prove that $z_i = 0$, for every $i = 1, \dots, n-2$. Namely, in this case $\rho_{i+1}(M) = \rho_i(M) + \frac{2\pi}{3}$ for every $1 \leq i \leq n-2$, and hence $\rho_{n-1}(M) = \rho_1(M) + \frac{2\pi}{3}(n-2)$. This implies $|\rho_{n-1}(M) - \rho_1(M)| \in \Omega(n)$, and thus $|\rho_1(M)| \in \Omega(n)$ or $|\rho_{n-1}(M)| \in \Omega(n)$.

Assume, for a contradiction, that $z_i \neq 0$, for some $1 \leq i \leq n-2$. If $z_i > 0$, then $\rho_{i+1}(M) \geq \rho_i(M) + \frac{8\pi}{3}$; further, if $z_i < 0$, then $\rho_{i+1}(M) \leq \rho_i(M) - \frac{4\pi}{3}$. Since each of these inequalities contradicts Lemma 6.2, the lemma follows. □

We are now ready to state the main theorem of this section.

Theorem 6.1 *There exists two straight-line planar drawings Γ_s and Γ_t of an n-vertex path P such that any planar morph between Γ_s and Γ_t requires $\Omega(n)$ morphing steps.*

Proof: The two drawings Γ_s and Γ_t of path $P = (v_1, \dots, v_n)$ are those illustrated in Figure 6.1. By Lemma 6.3, there exists an edge e_i of P, for some $1 \leq i \leq n-1$, such that $|\sum_{j=1}^{x-1} \rho_i^j| \in \Omega(n)$. Since, by Lemma 6.1, we have that $|\rho_i^j| < \pi$ for each $j = 1, \dots, x-1$, it follows that $x \in \Omega(n)$. This concludes the proof. □

Chapter 7

Morphing Planar Graph Drawings

In this chapter[1] we prove that, for any two drawings Γ_s and Γ_t of a plane n-vertex graph G, $O(n^2)$ planar linear morphing steps are always sufficient to construct a planar morph $\langle \Gamma_s, \dots, \Gamma_t \rangle$ transforming Γ_s into Γ_t, as stated in the following theorem.

Theorem 7.1 *Let Γ_s and Γ_t be two drawings of the same plane graph G. There exists a morph $\langle \Gamma_s, \dots, \Gamma_t \rangle$ with $O(n^2)$ steps transforming Γ_s into Γ_t .*

The chapter is structured as follows. In Section 7.1 we give an overview of the algorithm, which, as described in Section 7.2, proceeds by contracting a candidate vertex v onto a specific neighbor, by running two subroutines to handle the polygon induced by the neighbors of v in the cases in which they induce a quadrilateral (Section 7.3) or a pentagon (Section 7.4).

7.1 Overview of the Algorithm

In this section we give an high-level description of Algorithm `Morph` that recursively transforms Γ_s into Γ_t. Algorithm `Morph` proceeds as follows.

In the base case, graph G has a unique vertex v. Then, with a single linear morphing step, vertex v is moved from its position in Γ_s to its position in Γ_t (lines 1–2).

In each recursive step, a candidate vertex v of G, whose existence is guaranteed by Lemma 4.7, and a neighbor x of v are selected (lines 4–5), and the aim is to contract v

[1]The contents of this chapter are joint work with Soroush Alamdari, Patrizio Angelini, Timothy M. Chan, Giuseppe Di Battista, Fabrizio Frati, Anna Lubiw, Maurizio Patrignani, Sahil Singla, and Bryan T. Wilkinson, appeared in [AAC$^+$13], and have been submitted to journal.

Algorithm `Morph` (G, Γ_s, Γ_t)

Require: Γ_s and Γ_t are two planar drawings of a connected plane graph G

```
    /* base case */
 1. if $G$ has exactly one vertex $v$ then
 2.     $M \leftarrow$ linear morph from the position of $v$ in $\Gamma_s$ to the position of $v$ in $\Gamma_t$
 3.     return $M$
    /* begin of the recursive step */
 4. $v \leftarrow$ find a candidate vertex $(G)$
 5. $x \leftarrow$ find neighbor $(v, G)$
    /* morph $\Gamma_s$ and $\Gamma_t$ into drawings of $G$ where $v$ is
    $x$-contractible */
 6. $M_s \leftarrow$ make contractible $(v, x, \Gamma_s)$
 7. $M_t \leftarrow$ make contractible $(v, x, \Gamma_t)$
 8. $\Gamma_s^x \leftarrow$ get last drawing $(M_s)$
 9. $\Gamma_t^x \leftarrow$ get last drawing $(M_t)$
    /* pseudo-morph $\Gamma_s^x$ into $\Gamma_t^x$ */
10. $\Gamma'_s \leftarrow$ contract $(v, x, \Gamma_s^x)$
11. $\Gamma'_t \leftarrow$ contract $(v, x, \Gamma_t^x)$
    /* $G'$ is the graph obtained by the above contractions */
12. $\mathcal{M}_r \leftarrow$ Morph$(G', \Gamma'_s, \Gamma'_t)$
    /* construct the morph from $\Gamma_s$ to $\Gamma_t$ */
13. $M_r \leftarrow$ extend to actual morph$(\Gamma_s^x, \mathcal{M}_r, \Gamma_t^x)$
14. $\overline{M}_t \leftarrow$ reverse$(M_t)$
15. $M \leftarrow$ concatenate $(M_s, M_r, \overline{M}_t)$
16. return $M$
```

onto x. Note that vertex v might be not x-contractible in Γ_s or in Γ_t. Hence, in order to contract v onto x, drawings Γ_s and Γ_t are morphed into two suitable drawings Γ_s^x and Γ_t^x, respectively, in which v is x-contractible (lines 6–9). A detailed description of how to perform these morphs with $O(n)$ steps is given in Section 7.2. Denote by M_s and M_t morphs $\langle \Gamma_s, \dots, \Gamma_s^x \rangle$ and $\langle \Gamma_t, \dots, \Gamma_t^x \rangle$, respectively.

Since v is x-contractible in both Γ_s^x and Γ_t^x, vertex v is contracted onto x, thus obtaining two drawings Γ'_s and Γ'_t of a graph $G' = G/(v, x)$ with $n - 1$ vertices (lines 10–11).

Then, by applying a recursive call to the obtained graph and drawings (line 12), the algorithm computes a morph $\mathcal{M}_r$ of G', transforming Γ'_s into Γ'_t.

Note that, by definition, the morph $\mathcal{M}_r = \langle \Gamma'_s, \dots, \Gamma'_t \rangle$ of $G' = G/(v, x)$, together with the contraction of v onto x in Γ^x_s and the uncontraction of v from x in Γ'_t resulting in Γ^x_t, is a pseudo-morph of G from Γ^x_s to Γ^x_t. The uncontraction of v from x in Γ'_t is always possible, as the two drawings only differ for the position of v and x lies on the boundary of the kernel of v in Γ^x_t.

Hence, a planar linear morph M_r of Γ^x_s into Γ^x_t can be obtained from such a pseudo-morph by replacing the contraction and the uncontraction of v each with a linear morphing step and finding a suitable position for v at each time instant of $\mathcal{M}_r$ with the technique described in Section 4.6 (line 13).

Finally, a morph of Γ_s into Γ_t is obtained by concatenating M_s, M_r, and the reverse morph $\overline{M}_t = \langle \Gamma^x_t, \dots, \Gamma_t \rangle$ of $M_t = \langle \Gamma_t, \dots, \Gamma^x_t \rangle$ (lines 14–15).

7.1.1 Computing the Total Number of Steps

We claim that the algorithm has $T(n) \in O(n^2)$ steps. Namely, as we will show in Section 7.2, $O(n)$ steps suffice to construct morphs of Γ_s and Γ_t into drawings Γ^x_s and Γ^x_t of G, respectively, such that v is x-contractible onto the same neighbor x in both Γ^x_s and Γ^x_t. Further, two steps are sufficient to contract v onto x in both Γ^x_s and Γ^x_t, obtaining drawings Γ'_s and Γ'_t, respectively. Finally, the recursion on Γ'_s and Γ'_t takes $T(n-1)$ steps. Thus, $T(n) = T(n-1) + O(n) \in O(n^2)$. As described in Section 4.6, obtaining a morph from a pseudo-morph does not require any additional morphing step.

7.2 Making a Candidate Vertex x-Contractible

Let v be a candidate vertex of a plane graph G and let x be any neighbor of v. In this section we show how to construct a morph M with $O(n)$ steps transforming any straight-line planar drawing Γ of G into a straight-line planar drawing Γ^x of G in which v is x-contractible. Figure 7.1 shows two examples of drawings in which v is not x-contractible.

Assume that v is not x-contractible in Γ, as otherwise $\Gamma^x = \Gamma$ and no morph is required. This assumption implies that $\deg(v) \geq 2$. Namely, if v has degree 1, then it is always contractible onto its unique neighbor x in Γ.

In order to morph Γ into a straight-line planar drawing Γ^x of G in which v is x-contractible, we proceed as follows.

In a first step, which is described in detail in Section 7.2.1, denoting $S = G$ and $\Sigma = \Gamma$, we process Σ and S with $O(n)$ morphing steps and with a constant number

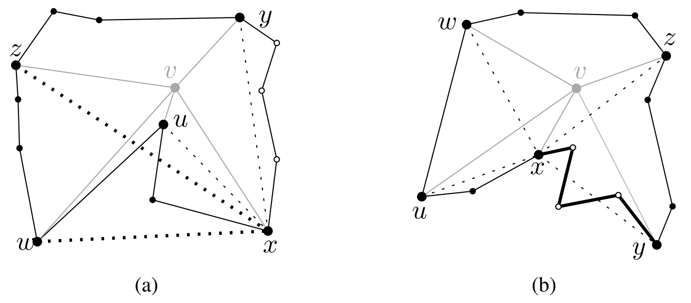


Figure 7.1: Two drawings in which v is not x-contractible.

of edge additions, in order to obtain a planar straight-line drawing Σ_v of a supergraph S_v of S with the following properties: (i) there is an edge between each pair of consecutive neighbors of v, and (ii) v is a candidate vertex of S_v. By definition of candidate vertex, all the faces incident to v in S_v are simple and empty. As also observed by Cairns [Cai44a, Lemma 3.1] (and proven in Lemma 4.3), this implies that at least one of the neighbors of v, say u, lies on the boundary of the kernel of v in Σ_v.

In a second step, which is described in detail in Section 7.2.2, we exploit the property that v is u-contractible to compute a morph with $O(n)$ steps transforming Σ_v into a drawing Σ^x in which v is x-contractible. Finally, a morph transforming Γ into Γ^x with $O(n)$ steps is obtained by restricting the morph of Σ into Σ^x to the vertices and the edges of G.

7.2.1 Connecting Consecutive Neighbors of v

Let u and w be two consecutive neighbors of v and let $\Delta = \langle u, v, w\rangle$ be the closed triangular region bounded by these three vertices in the current drawing Σ of S. Our purpose is to add edge (u, w) while maintaining v a candidate vertex, that is, in such a way that cycle (u, v, w) bounds a simple face of S.

Assume that adding edge (u, w) to Σ and S results in a non-planar drawing or in a planar drawing in which v is not a candidate vertex of S. It follows that some vertices of S, different from u, v, and w lie inside Δ, as depicted in Figure 7.2(a). In order to "push away" such vertices from Δ, we proceed as follows.

Let Σ_r be the drawing of a plane graph S_r obtained by adding a new vertex r and the edges (r, v), (r, u), and (r, w) to Σ and to S, in such a way that Σ_r is straight-line planar and cycles (v, r, u) and (v, r, w) bound two simple faces of S_r.

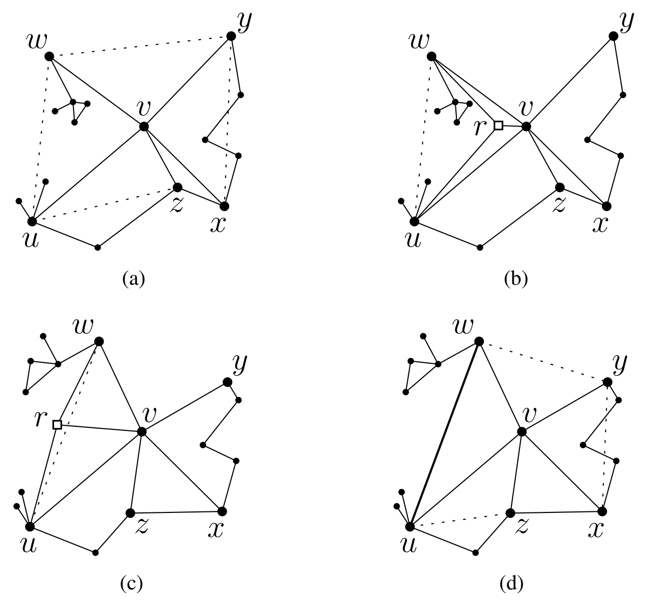
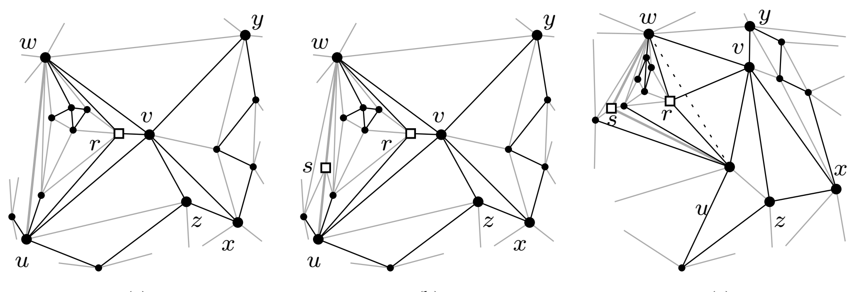


Figure 7.2: Vertex v and its neighbors. (a) Vertices u and w do not have direct visibility and the triangle $\langle u, w, v\rangle$ is not empty. (b) A vertex r is added suitably close to v and connected to v, u, and w. (c) The output of `AlgoQuad` on the quadrilateral $\langle u, r, w, v\rangle$. (d) Vertex r and its incident edges can be removed in order to insert edge (u, w).

With a technique similar to that described in Section 4.3, based on the proof of Fáry's Theorem [Fár48], a position for r with such properties can be found in Σ, suitably close to v. See Figure 7.2(b) for an example.

Augment Σ_r to the drawing Θ of a maximal plane graph T by first adding three vertices a, b, and c to Σ_r, so that triangle $\langle a, b, c\rangle$ completely encloses the rest of the drawing, and then adding dummy edges [Cha91] till a maximal plane graph is obtained.

It might be the case that edge (u, w) has been added to S_r during this augmentation. Observe that edge (u, w) prevents the existence of a planar morph of Θ to a drawing of T in which Δ only contains its delimiting vertices. In this case, we subdivide (u, w) in Θ (namely, replace edge (u, w) with edges (s, w) and (s, u), placing s along the straight-line segment connecting u and w) and triangulate the two faces vertex s is incident to. This case is depicted in Figure 7.3.



(a) (b) (c)

Figure 7.3: If edge (u, w) has been added when augmenting S_r to T (a), then we subdivide (u, w) with the insertion of a vertex s and we triangulate the two faces s is incident to (b). (c) The output of `AlgoQuad` on the quadrilateral $\langle u, r, w, v\rangle$.

By applying algorithm `AlgoQuad` to Θ and the quadrilateral $\langle u, v, w, r\rangle$ we morph Θ with a linear number of steps into a drawing Θ' of T in which $\langle u, v, w, r\rangle$ is convex (see Figures 7.2(c) and 7.3(c)). Algorithm `AlgoQuad` is completely described in Section 7.3.

Let Σ'_r be the drawing of S_r obtained by restricting Θ' to the vertices and the edges of S_r. Since $\langle u, v, w, r\rangle$ is a convex polygon containing no vertex of S_r in its interior, vertex r and its incident edges can be removed and edge (u, w) can be added to S and to Σ'_r, so that the resulting drawing Σ' is planar and cycle (u, w, v) bounds a simple face of S (see Figure 7.2(d)).

Once edge (u, w) has been added to S, if $\deg(v) = 2$ then v is both u-contractible and w-contractible. Otherwise, apply the same operations described before for every other pair of consecutive neighbors of v, until all faces incident to v are triangular while v is a candidate vertex.

7.2.2 Moving x to the Boundary of the Kernel of v

Let Σ_v be the current drawing of S obtained after applying the procedure described in Section 7.2.1. If $\deg(v) \leq 3$, then v is x-contractible for any neighbor x, i.e. every neighbor x of v lies on the boundary of the kernel of v in Σ_v. Otherwise, it may be the case that v is not x-contractible. If that's the case, add three vertices a, b, and c to Σ_v, so that triangle $\langle a, b, c\rangle$ completely encloses the rest of the drawing, and then add dummy edges [Cha91] till a drawing Θ of a maximal plane graph T is obtained.

In the following we show how to construct a morph M_Θ transforming, with a linear number of steps, Θ into a drawing Θ^x in which v is x-contractible.

If $\deg(v) = 4$, proceed as follows. Let u be a neighbor of v such that v is u-contractible. We contract v onto u in Θ obtaining a drawing Θ' of a maximal plane graph T'. We apply algorithm `AlgoQuad` to construct a morph with $O(n)$ steps transforming Θ' into a drawing Θ'' such that the quadrilateral induced by the neighbors of v in T is convex. Then, uncontract v from u, thus obtaining a drawing Θ^x of T in which all the neighbors of v (and hence also x) lie on the boundary of its kernel. Observe that the contraction of v onto u, followed by the morph of Θ' into Θ'' and by the uncontraction of v from u is a pseudo morph $\mathcal{M}_\Theta$ of Θ into Θ^x. Finally, M_Θ can be obtained from $\mathcal{M}_\Theta$ by applying the technique described in Section 4.6 with no additional morphing steps.

If $\deg(v) = 5$, a morph $M_\Theta = \langle \Theta, \dots, \Theta^x \rangle$ with $O(n)$ steps is obtained by applying algorithm `AlgoPenta`, described in Section 7.4, to Θ, v, and x.

This concludes the description of the algorithm, as $\deg(v) \leq 5$, given that v is a candidate vertex.

7.3 Algorithm **`AlgoQuad`**

The input of algorithm `AlgoQuad` consists of:

- a planar straight-line drawing Υ of a maximal plane graph Y, and
- a set $\{x, y, z, w\}$ of vertices of Y inducing a biconnected outerplane graph Q not containing any other vertex in its interior in Υ.

The output of algorithm `AlgoQuad` is a planar morph with $O(n)$ linear morphing steps transforming Υ into a drawing Υ^* of Y in which the vertices of Q induce a convex quadrilateral.

Assume that Q is not convex in Υ, as otherwise $\Upsilon^* = \Upsilon$. Observe that, since Q is outerplane, graph Y cannot contain both edge (w, x) and edge (y, z) but, since Y is maximal plane, it contains exactly one of them. It follows that either vertex y or vertex z lies in the triangular region delimited by the remaining three vertices. In the following we assume that edge (y, z) belongs to Y and that vertex y lies in the interior of the triangle delimited by x, w, and z, as depicted in Figure 7.4.

We follow Cairns's idea [Cai44a], completely described in Section 3.2, of computing a pseudo-morph $\mathcal{M}$ transforming Υ into a drawing Υ^* (in which the vertices of Q delimit a convex quadrilateral) by contracting an internal vertex with degree at most 5 onto one of its neighbors and then recursively compute a pseudo-morph of the resulting graph.

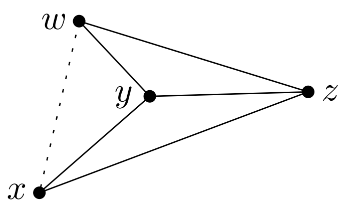


Figure 7.4: Since edge (y, z) belongs to Y, $(w, x) \notin Y$, as otherwise Q would not be outerplane. Vertex v lies in the interior of the triangle delimited by x, w, and z.

However, in order to achieve the convexity of Q in Υ^*, it is important to avoid introducing edge (w, x) or destroying Q itself when contracting a vertex of Y onto one of its neighbors. Also, as in Cairns's approach, we contract internal vertices only. Denote by a, b, and c the three vertices delimiting the external face of Y and observe that since edge (w, x) does not belong to Y, at most two vertices among x, w, and z can be incident to the outer face of Y. We say that a vertex v of Y is *problematic* if it is of one of the following three types:

Type 1: v belongs to Q and is not incident to the external face of Y; or

Type 2: $v \in \{a, b, c\}$, i.e., it is incident to the external face of Y; or

Type 3: v is an internal vertex of Y, $\deg(v) \leq 5$, edges (v, w) and (v, x) belong to Y, and v is either w- or x-contractible.

We call a vertex v of the third type a xw-inducing vertex, as the contraction of v onto either of w or x would induce the external chord (w, x) of Q. Also, we define the *deficiency* $\text{def}(v)$ of a vertex v to be $6 - \deg(v)$. By Euler's formula for planar graphs, we have that $\sum_{v \in Y} \text{def}(v) = 6n - \sum_{v \in Y} \deg(v) = 12$.

If Y contains at least a candidate vertex that is non-problematic, Cairns's approach works fine. In the following, we show that either Y contains a problematic vertex that can be handled, or it contains a non-problematic internal vertex with degree at most 5. Namely, in the first case we can either morph directly to convexify Q, or we can perform a contraction that reduces the problem size by one, while, in the second case, we bound the deficiencies of the problematic vertices that cannot be directly handled by using a counting argument.

7.3.1 Handleable Problematic Vertices of Type 1

Recall that problematic vertices of type 1 belong to quadrilateral Q and are not incident to the external face of Y. Let v be a problematic vertex of Type 1. In the following, we show that if $\deg(v) \leq 4$, it can be directly handled, while the case in which $\deg(v) \geq 5$ is considered in Section 7.3.4. We distinguish three cases:

Case 1.1: $v = w$, or $v = x$. We only describe the case in which $v = w$ (see Figure 7.5), being the case $v = x$ symmetric.

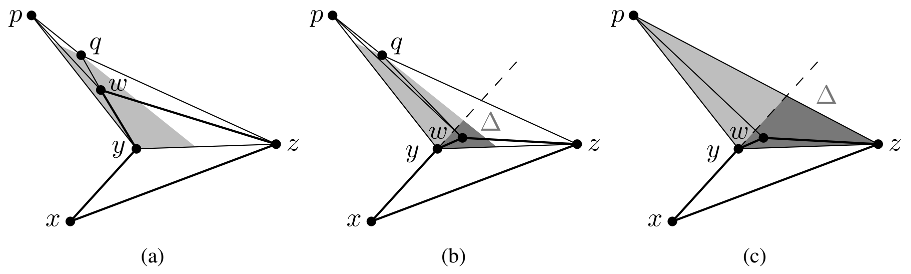


Figure 7.5: If $\deg(w) \leq 4$, the problematic vertex w can be handled. (a) Either y or z lie on the boundary of the kernel of w (grey region). (b) Quadrilateral Q can be directly convexified by moving w to any point of Δ (dark grey region). (c) If $\deg(w) = 3$, the same argument applies.

Observe that since $\deg(v) \leq 4$, by Lemma 4.3, either y or z, say y, lies on the boundary of the kernel of w.

Consider the elongation, emanating from vertex y, of edge (x, y) and note that it traverses the kernel of w. Further, since the kernel of w is partially delimited by edge (y, z), the intersection Δ between the kernel of w and the wedge delimited by the elongation of (x, y) and edge (y, z) is non empty.

Finally, note that any point w' of Δ is such that the quadrilateral (x, y, w', z) is convex. Then, Q can be directly convexified, with a unique planar linear morphing step, by moving w to any point w' of Δ. Note that, if $\deg(w) = 3$, the same argument applies (see Figure 7.5(c)).

Case 1.2: $v = z$. Since z has three neighbors in Q and is not incident to the external face of Y, it has at least one neighbor not in Q. We show that z can be handled if $\deg(v) = 4$, while the case in which $\deg(v) \geq 5$ is considered in

Section 7.3.4. Let then p be the unique neighbor of z not in Q. Since Y is a maximal plane graph, edges (p, w) and (p, x) belong to Y. This situation is depicted in Figure 7.6.

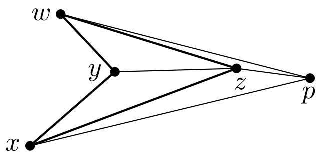


Figure 7.6: If $\deg(z) = 4$, then it is contractible onto its unique neighbor not in Q.

Analogously to the previous case, since two of the neighbors of z lie on the boundary of its kernel and since the straight-line segment $\overline{wx}$ does not lie in the interior of the quadrilateral induced by x, y, w, and p, vertex z is p-contractible. Then, we contract z onto p, recursively convexify the quadrilateral induced by x, y, w, and p and then uncontract z, thus convexifying Q.

Case 1.3: $v = y$. As above, we consider only the case in which $\deg(v) = 4$. Let p the unique neighbor of y not in Q. If p does not lie in the interior of the triangular region delimited by x, w, and z (see Figure 7.7(a)), then Q can be directly convexified by moving y to any point y' inside the triangular region bounded by p, x, and w. Otherwise, namely vertex p lies inside the triangular region delimited by x, w, and z (see Figure 7.7(b)), as in **Case 1.2**, we contract y onto p, recursively convexify the quadrilateral delimited by x, p, w, and z, and then uncontract y, thus convexifying Q.

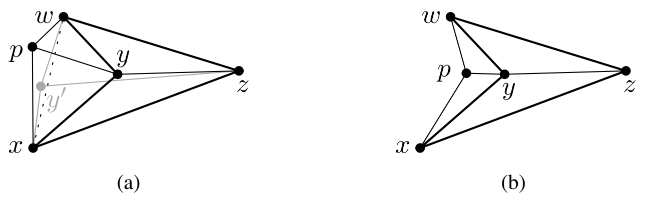


Figure 7.7: If $\deg(y) = 4$, either quadrilateral Q can be convexified by moving y to any point y' inside triangle xpw (a), or it can be contracted onto p (b).

We have shown that if the vertices of Q have degree at most 4 and are not incident to the external face of Y, can be directly handled. So assume that they all have degree at least 5, i.e., deficiency at most 1, for a total of 4. This case is considered in Section 7.3.4.

7.3.2 Handleable Problematic Vertices of Type 2

Recall that problematic vertices of type 2 are the vertices, denoted by a, b, and c, incident to the external face of the graph. We show that such vertices can be handled if one of them, say a, has degree 3, or if all of them have degree 4. Observe that in the case in which all the external vertices have degree 3, the graph does not contain any quadrilateral, as it is the complete graph on four vertices.

Case 2.1: $\deg(a) = 3$. Since Y is a maximal planar graph, (i) vertex a has a neighbor p that is also adjacent to b and c (see Figure 7.8(a)), and (ii) triangles apb and apc bound two faces of Y.

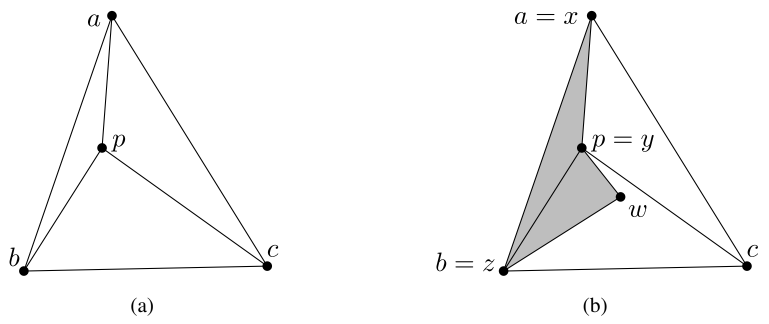


Figure 7.8: (a) If $\deg(a) = 3$, then a, b, and c have a common neighbor p. (b) If Q (grey area) has an edge on the external face, then it has exactly one vertex in triangle pbc.

If Q lies entirely inside triangle pbc, then we can simply recursively morph the subgraph $Y' = Y \setminus \{a\}$, having p, b, and c on the external face. Otherwise, some vertices of Q are incident to the external face of Y. Observe that, since -by hypothesis- edge (w, x) does not belong to Y, they are exactly two. Also, Q is composed of either apb or acp, say apb, and an adjacent triangle in pbc. It follows that vertices y and p, a and x, and b and z coincide, while w lies inside pbc (see Figure 7.8(b)). Let s be a straight-line segment and denote by $sl(s)$ the slope of s. We have that $sl(\overline{pc}) < sl(\overline{pw}) < sl(\overline{pb})$. Denote by Υ the current drawing of Y.

Observe that, by fixing the barycentric coordinates of the vertices of Y lying inside triangle pbc with respect to such vertices and applying the technique described in Section 4.4, we can morph Υ to any drawing in which p is placed at any internal point of abc with a unique planar linear morphing step.

As in the proof of Fáry's Theorem [Fár48], there exists a disk D_x, centered at vertex x, such that x placing x at any internal point of D_x while maintaining any other vertex of Y at the same position it has in Υ yields a straight-line planar drawing of Y. Let y' be any point arbitrarily close to $a = x$ in the intersection between D_x and triangle abc and observe that, by fixing the barycentric coordinates of the vertices of Y lying inside triangle pbc with respect to such vertices and applying the technique described in Section 4.4, we can morph Υ to a drawing Υ' of Y in which p is placed at point y' with a unique planar linear morphing step.

Observe that, since in Υ' a lies on the boundary of the kernel of y and since y lies inside triangle abc edge (x, c) delimits the boundary of the kernel of y (see Figure 7.9(a)). Denote by Δ the triangular region delimited by the elongation of edge (z, w) emanating from w, edge (x, c), and the straight-line segment $\overline{xw}$.

Then, with an additional planar linear morphing step, we directly convexify Q by moving vertex y only, from its position in Υ' to any point of Δ (Figure 7.9(b)).

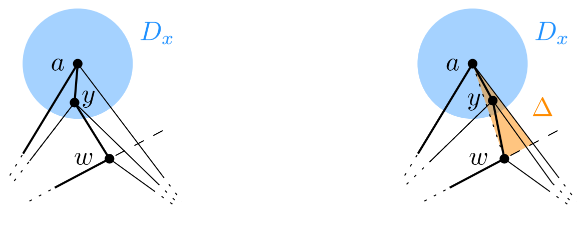


Figure 7.9: Detail of drawing Υ'. (a) Since y lies in D_x, its kernel is delimited by edge (x, c). (b) Q can be convexified by moving y to any point of Δ.

Case 2.2: $\mathbf{deg(a) = deg(b) = deg(c) = 4}$. Denote by p, q, and r the neighbors of a, b, and c and observe that all the remaining vertices of Y lie inside triangle pqr (see Figure 7.10). We distinguish three subcases, depending on the positions of the vertices of Q.

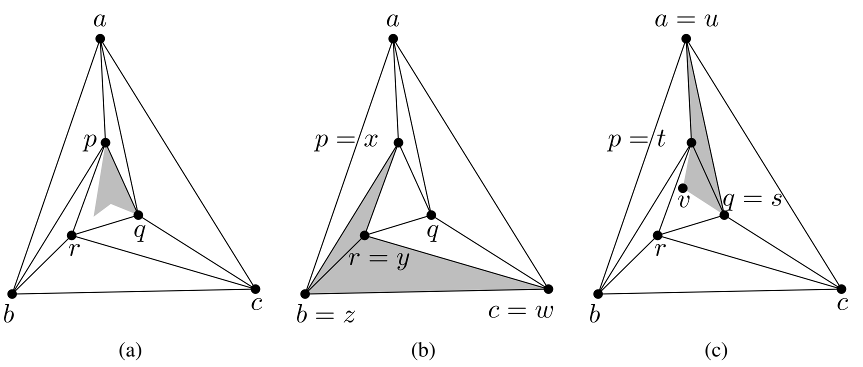


Figure 7.10: Possible configurations for the vertices of Q in Case 2.2. (a) No vertex of Q lies outside triangle pqr. (b) No vertex of Q lies inside pqr. (c) One vertex of Q lies inside and one outside pqr.

First, if no vertex of Q lies on the external face of Y (see Figure 7.10(a)), then we simply apply recursion on triangle pqr.

Second, if no vertex of Q lies inside pqr, then Q is composed of two adjacent internal faces of Y (see Figure 7.10(b)) and can be convexified in a constant number of steps. Namely, while maintaining fixed the barycentric coordinates of the vertices lying inside pqr (see Section 4.4), it suffices to simulate the rotation of triangle pqr (see Section 4.5).

Third, Q is composed of a vertex u incident to the external face of Y, a vertex v lying inside triangle pqr, and two vertices (denoted by s and t) of pqr (see Figure 7.10(c)). Observe that, since the $sl(vs)$ and $sl(vt)$ are bounded by the slopes of the two edges of Q different from (s,t), convexifying the polygon induced by p, q, r, and u necessarily also convexifies Q. As in the previous case, we perform this operation by fixing the barycentric coordinates of the vertices lying inside pqr and simulating the rotation of such a triangle.

We have shown that if one of the external vertices of Y has degree 3, or if all of them have degree 4, these problematic vertices can be easily handled. So assume that each of a, b, and c has degree at least 4 and that one of them has degree at least 5, i.e. the sum of their deficiencies is at most 5. This case is considered in Section 7.3.4.

7.3.3 Handleable Problematic Vertices of Type 3

Recall that a problematic vertex v of Type 3, called xw-inducing vertex, is an internal vertex v of Y with degree at most 5 such that edges (v, w) and (v, x) belong to Y, and v is either w- or x-contractible. See Figure 7.11(a).

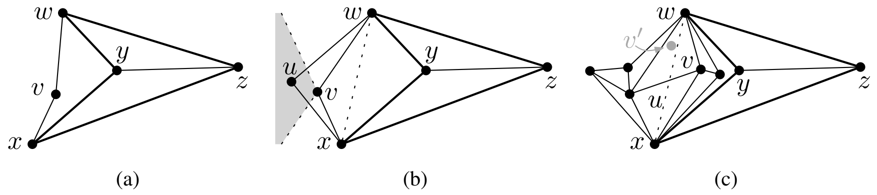


Figure 7.11: (a) An xw-inducing vertex v. (b) Illustration for Lemma 7.1. Vertex u cannot induce chord (x, w) since neither x nor w lie on the boundary of its kernel (grey region). (c) Moving v to v' makes u a non-problematic vertex.

We first prove that graph Y can contain at most two xw-inducing vertices.

Lemma 7.1 *Graph Y contains at most two xw-inducing vertices.*

Proof: We prove the statement by showing that two xw-inducing vertices cannot lie on the same side of segment $\overline{xw}$, thus implying the statement. Let then u and v be two xw-inducing vertices and assume, for a contradiction, that they lie on the same side of $\overline{xw}$ (see Figure 7.11(b)). Consider the quadrilateral K induced by u, w, v, and x in Υ, and x and observe that segment $\overline{xw}$ does not lie in its interior. Assume without loss of generality that v is nearer than u to segment $\overline{xw}$. This implies that neither x nor w can lie on the boundary of the kernel of u, contradicting the necessary condition that u is either w- or x-contractible for u being a xw-inducing vertex. □

In the following we show that, if Y contains two xw-inducing vertices u and v, we can modify the current drawing Υ of Y and contract one of them without introducing edge (x, w).

Assume, without loss of generality, that v lies inside the triangular region delimited by x, y, and w. Since u and v cannot lie on the same side of segment $\overline{xw}$, vertex u lies outside such a region. Also, since v is an xw-inducing vertex, it is either w- or x-contractible. Assume the latter.

Then, by the proof of Fáry's Theorem [Fár48] and by the fact that the quadrilateral delimited by u, w, v, and x cannot contain any vertices, there exists a point v' of the kernel of v that is arbitrarily close to x and lies inside the triangular region delimited by u, x, and w (see Figure 7.11(c)). Then with a unique planar linear morphing step, we move vertex v to point v', while leaving all the remaining vertices at their positions. Denote by Υ' the obtained drawing of Y and observe that, in Υ', both u and v lie on the same side of $\overline{xw}$. By the proof of Lemma 7.1, it follows that u is not an xw-inducing vertex any more. Also, since $\deg(u) \leq 5$, there exists a neighbor p of u, different from both x and w, that lies on the boundary of the kernel of u. So, we can contract u onto p and recursively convexify Q.

As we can repeatedly apply such a preocedure, we can assume that Y contains at most one xw-inducing vertex. Since it has degree at least 4, its deficiency is at most 2. This case is considered in Section 7.3.4.

7.3.4 Existence of a non-problematic vertex

In the following we show that, if none of the cases described in Sections 7.3.1 to 7.3.3 applies, then Y contains a non-problematic vertex.

Denote by d_i, with $i \in \{1, 2, 3\}$ the sum of the deficiencies of the problematic vertices of Type i. If none of the cases described in Sections 7.3.1 to 7.3.3, we have that: $d_1 \leq 4$, $d_2 \leq 5$, and $d_3 \leq 2$. Hence, since by Euler's formula for planar graphs the sum d_Y of the deficiencies of the vertices of Y equals 12, while $d_1 + d_2 + d_3 \leq 11 < d_Y = 12$, graph Y contains a non-problematic vertex v with degree at most 5 that, by Lemma 4.3, can be contracted onto some of its neighbors in order to recursively convexify quadrilateral Q.

7.3.5 Computing the Total Number of Planar Linear Morphing Steps

Let $T_{AQ}(n) = T_{AQ}(n-1) + O(1)$ be the total number of planar linear morphing steps required by `AlgoQuad`. In Sections 7.3.1 to 7.3.4 we have shown that with a constant number of planar linear morphing steps either the quadrilateral Q is directly convexified, or the size of the input graph is reduced by one. It follows that $T_{AQ} \in O(n)$.

7.4 Algorithm `AlgoPenta`

The input of algorithm `AlgoPenta` consists of:

- a planar straight-line drawing Υ of a maximal plane graph Y;

- a contractible vertex v of Y; and
- a neighbor x of v.

The output of algorithm `AlgoPenta` is a planar morph M_Υ of $O(n)$ linear morphing steps transforming Υ into a drawing Υ_x of Y in which vertex x lies on the boundary of the kernel of v, and hence v is x-contractible.

Let x, y, z, u, and w be the neighbors of v, appearing in this order, in Y. Since v is contractible in Y, at least one of its neighbors, denoted by r, lies on the boundary of the kernel of v in Υ. We distinguish two cases:

(a) vertices x and r are non-consecutive neighbors of v in Y (see Figure 7.12(a)); and

(b) vertices x and r are consecutive neighbors of v in Y (see Figure 7.12(b)).

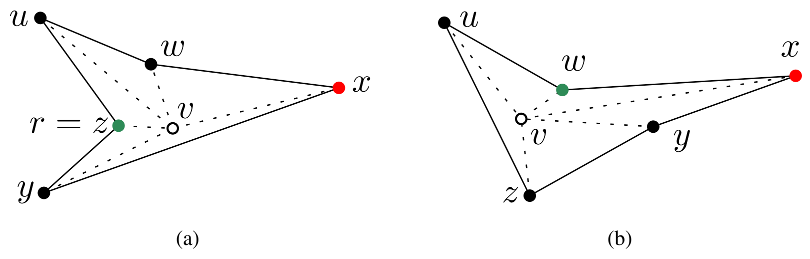


Figure 7.12: Possible input cases for `AlgoPenta`. (a) Vertices x and r are non-consecutive neighbors of v. (b) Vertices x and r are consecutive neighbors of v.

In both cases, the idea is that of reducing the problem of making v x-contractible to the problem of convexifying a quadrilateral, and hence apply algorithm `AlgoQuad`. The linear morph $M_\Upsilon = \langle \Upsilon, \dots, \Upsilon_x \rangle$ is computed as follows.

Case (a): vertices r and x are non-consecutive neighbors of v in Y

Refer to Figure 7.12(a) and assume without loss of generality that $r = z$, the case in which $r = u$ being analogous.

Observe that, since edges (x, y) and (z, y) belong to Y, any planar drawing of Y in which quadrilateral $\mathcal{C} = (z, u, w, x)$ is convex is such that both x and z lie on the boundary of the kernel of v (see Figure 7.13). Also, the existence of such drawings is guaranteed by the fact that neither edge (u, x), nor edge (u, y) belong to Y.

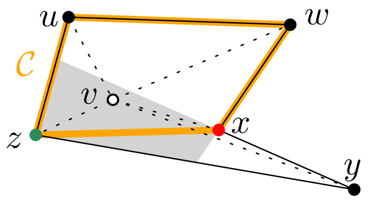


Figure 7.13: Any planar straight-line drawing of Y in which $\mathcal{C} = (z, u, w, x)$ (orange quadrilateral) is convex is such that both x and z lie on the boundary of the kernel of v (grey region).

In order to obtain a drawing (that we denote by Υ_x) fulfilling such a property, we first contract v onto z in Υ, thus obtaining a drawing Υ' of graph $Y' = Y/(v, z)$, and then apply algorithm `AlgoQuad` to Υ' and $\mathcal{C}$.

Denote by $M' = \langle \Upsilon', \dots, \Upsilon'_x \rangle$ the morphing sequence with $O(n)$ steps obtained by applying algorithm `AlgoQuad` to Υ' and $\mathcal{C}$ and note that the contraction of v onto z in Υ, followed by M' and by the uncontraction of v from z in Υ'_x yielding drawing Υ_x, is a pseudo-morph $\mathcal{M}$ of Y transforming Υ into Υ_x.

The final morph $M_\Upsilon = \langle \Upsilon, \dots, \Upsilon_x \rangle$ can be obtained, with no additional morphing step, by applying the technique described in Section 4.6 to $\mathcal{M}'$.

Case (b): vertices r and x are consecutive neighbors of v in Y

Refer to Figure 7.12(b) and assume without loss of generality that $r = w$, the case in which $r = y$ being analogous. We apply algorithm `AlgoQuad` to reduce to the previous case, as follows.

Consider the quadrilateral $\mathcal{C}_{wz} = (x, y, z, w)$ and observe that any drawing of Y in which $\mathcal{C}_{wz}$ is convex is such that both w and z lie on the boundary of the kernel of v (see Figure 7.14). Also, the existence of such drawings is guaranteed by the fact that neither edge (u, x), nor edge (u, y) belong to Y.

In order to obtain a drawing, denoted by Υ_{wz}, fulfilling such a property, we first contract v onto w in Υ, thus obtaining a drawing Υ' of graph $Y' = Y/(v, w)$, and then apply algorithm `AlgoQuad` to Υ' and $\mathcal{C}_{wz}$.

Denote by $M'_{wz} = \langle \Upsilon', \dots, \Upsilon'_{wz} \rangle$ the morphing sequence with $O(n)$ steps obtained by applying algorithm `AlgoQuad` to Υ' and $\mathcal{C}_{wz}$ and note that the contraction of v onto w in Υ, followed by M' and by the uncontraction of v from w in Υ'_{wz} yielding drawing Υ_{wz}, is a pseudo-morph $\mathcal{M}_{wz}$ of Y transforming Υ into Υ_{wz}. A

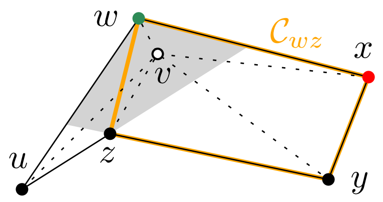


Figure 7.14: Any planar straight-line drawing of Y in which $\mathcal{C}_{wz} = (x, y, z, w)$ (orange quadrilateral) is convex is such that both z and w lie on the boundary of the kernel of v (grey region).

morph $M_{wz} = \langle \Upsilon, \ldots, \Upsilon_{wz} \rangle$ can be obtained, with no additional morphing step, by applying the technique described in Section 4.6 to $\mathcal{M}_{wz}$.

Since vertex z lies on the boundary of the kernel of v in Υ_{wz}, the previous case applies. Let M_{zx} be the morph with $O(n)$ steps obtained by applying the previous case to Υ_{wz} and quadrilateral $\mathcal{C}_{zx} = z, u, w, x$, and transforming Υ_{wz} into a drawing Υ_x in which v is x-contractible.

Morph $M_\Upsilon = \langle \Upsilon, \ldots, \Upsilon_x \rangle$ is finally obtained by concatenating M_{wz} and M_{zx}.

Chapter 8

Conclusions and Open Problems

In Chapter 5 we provided algorithms to compute planar morphs of n-vertex series-parallel graphs with $O(n)$ steps. Due to the lower bound proved in Chapter 6, such algorithm is asymptotically optimal. Further, in Chapter 7, we focused on general plane graphs and provided an algorithm, that is currently the most efficient, to compute planar morphs with $O(n^2)$ steps.

In the following we propose some open problems on this topic.
The first natural problem is motivated by the linear upper bound appeared in [ADD$^+$14].

Open Problem 8.1 *Given two planar straight-line drawings Γ_s and Γ_t of a plane graph, can we efficiently compute the minimum number of steps required to morph Γ_s into Γ_t?*

The algorithms provided in this part extensively exploits edge contractions, resulting in a poor resolution and large area (i.e, exponential) required by intermediate drawings during the morph. This is due to the fact that the contraction of an edge is simulated by keeping its endpoints exponentially close to each other with respect to their distance in the two input drawings.

Open Problem 8.2 *Given any two planar straight-line drawings Γ_s and Γ_t on the $n \times n$ grid, does there exists a morph whose intermediate drawings require polynomial area?*

All the known algorithms to morph graph drawings only apply to planar graphs. A natural generalization would be to study of morphs of non-planar graph drawings.

Open Problem 8.3 *Given any two straight-line drawings Γ_s and Γ_t in which the same pairs of edges cross in both drawings, does there exist a morph $\langle \Gamma_s, \ldots, \Gamma_t \rangle$ with a polynomial number of steps in which no further crossings are introduced? If so, what is a lower bound on the number of steps?*

Part II

Visiting Drawings of Graphs

Chapter 9

Monotone Drawings of Graphs with Fixed Embedding

A drawing of a graph is a *monotone drawing* if for every pair of vertices u and v there is a path drawn from u to v that is monotone in some direction. In this chapter[1] we investigate planar monotone drawings in the *fixed embedding setting*, i.e., a planar embedding of the graph is given as part of the input that must be preserved by the drawing algorithm. In this setting we prove that every planar graph on n vertices admits a planar monotone drawing with at most two bends per edge and with at most $4n - 10$ bends in total; such a drawing can be computed in linear time and requires polynomial area. We also show that two bends per edge are sometimes necessary on a linear number of edges of the graph. Furthermore, we investigate subclasses of planar graphs that can be realized as embedding-preserving monotone drawings with straight-line edges. In fact, we prove that biconnected embedded planar graphs and outerplane graphs always admit such drawings, and describe linear-time drawing algorithms for these two graph classes.

9.1 Introduction

A drawing of a graph is a *monotone drawing* if for every pair of vertices u and v there is a path drawn from u to v that is monotone in some direction. In other words, a drawing is monotone if, for any given direction d (e.g., from left to right) and for each

[1]The contents of this chapter are joint work with Patrizio Angelini, Walter Didimo, Stephen Kobourov, Tamara Mchedlidze, Antonios Symvonis, and Stephen Wismath, and appeared in [ADK$^+$11, ADK$^+$13]

pair of vertices u and v, there exists a suitable rotation of the drawing for which a path from u to v becomes monotone in the direction d.

Monotone drawings have been recently introduced [ACB$^+$12] as a new visualization paradigm, which is well motivated by human subject experiments by Huang *et al.* [HEH09], who showed that the "geodesic path tendency" (paths following a given direction) is important in comprehending the underlying graph. Monotone drawings are related to well-studied drawing conventions, such as upward drawings [DETT99, GT95], greedy drawings [AFG10, LM10, PR05], and the geometric problem of finding monotone trajectories between two given points in the plane avoiding convex obstacles [ACM89].

Planar monotone drawings with straight-line edges form a natural setting and it is known that biconnected planar graphs and trees always admit such drawings, for some combinatorial embedding of the graph [ACB$^+$12]. Recently, Hossain and Rahman [HR13] proved that a series-parallel graph of n vertices has a straight-line planar monotone drawing on a grid of size $O(n) \times O(n^2)$.

However, the question whether a simply connected planar graph always admits a planar straight-line monotone drawing or not is still open.

On the other hand, in the *fixed embedding setting* (i.e., the planar embedding of the graph is given as part of the input and the drawing algorithm is not allowed to alter it) it is known [ACB$^+$12] that there exist simply connected planar embedded graphs that admit no straight-line monotone drawings.

In this chapter we study planar monotone drawings of graphs in the fixed embedding setting, answering the natural question whether monotone drawings with a given constant number of bends per edge can always be computed, and identifying some subclasses of planar graphs that always admit planar monotone drawings with straight-line edges. Our contributions are summarized below:

- We prove that every n-vertex plane graph has an embedding-preserving monotone drawing with *curve complexity* 2 and with at most $4n - 10$ bends in total. Such a drawing can be computed in linear time and requires polynomial area. We recall that the curve complexity is the maximum number of bends along an edge.

- We show that our bound on the curve complexity is tight, i.e., there exist infinitely many embedded planar graphs that do not admit any embedding-preserving monotone drawing with at most one bend per edge. More specifically, we prove that each of these graphs requires two bends on a linear number of edges.

- We investigate what subfamilies of embedded planar graphs can be realized as embedding-preserving monotone drawings with straight-line edges. We prove that outerplane graphs and biconnected embedded planar graphs always admit such a drawing, which can be computed in linear time.

The chapter is structured as follows. Basic definitions and results are given in Section 9.2. An algorithm for computing embedding-preserving monotone drawings of general embedded planar graphs with curve complexity 2 and with at most $4n - 10$ bends in total is described in Section 9.3, together with a proof that such a bound is tight. Algorithms for computing straight-line monotone drawings of meaningful subfamilies of embedded planar graphs are given in Section 9.4.

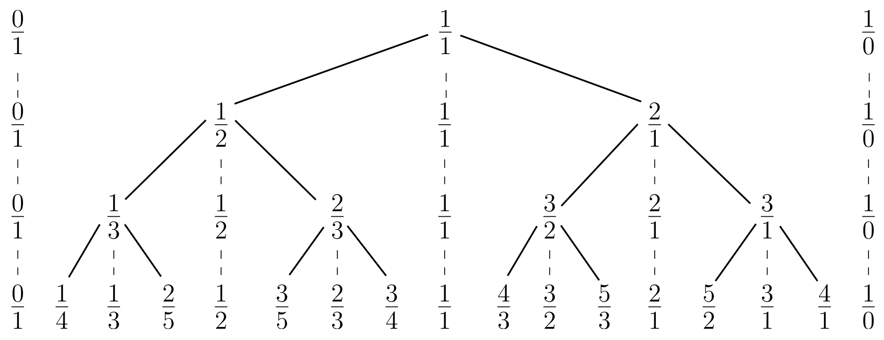
9.2 Preliminaries

Let Γ be a drawing of a graph. A path $u = u_1, \ldots, u_k = v$ between vertices u and v in Γ is *monotone* with respect to a direction d if the orthogonal projections of vertices $u_1, \ldots, u_k$ on d appear in the same order as the vertices appear in the path. Drawing Γ is *monotone* if for every pair of vertices u and v there exists a path $p(u, v)$ and a direction d such that $p(u, v)$ is monotone with respect to d.

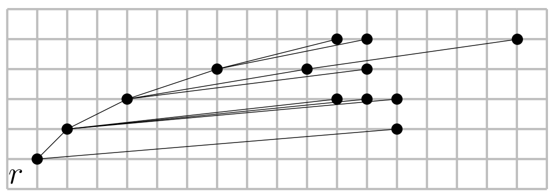
In [ACB$^+$12] it has been shown that every tree admits a straight-line monotone drawing in polynomial area. Namely, the authors provide two algorithms, called *Algorithm BFS-based* and *Algorithm DFS-based*, that construct drawings requiring $O(n^{1.6}) \times O(n^{1.6})$ and $O(n) \times O(n^2)$ area, respectively. Both algorithms rely on the concept of the *Stern-Brocot tree* [Ste58, Bro60] $\mathcal{SB}$, an infinite tree whose nodes are in bijective mapping with the irreducible positive rational numbers. The first four levels of the Stern-Brocot tree are depicted in Figure 9.1(a).

In particular, *Algorithm DFS-based* assigns slopes $\frac{1}{1}, \frac{2}{1}, \ldots, \frac{n-1}{1}$ to the edges of the input tree T according to the order given by a DFS-visit of T. Note that the assigned slopes correspond to the first $n-1$ elements of the rightmost path of $\mathcal{SB}$. The drawing of a tree obtained with Algorithm DFS-based is illustrated in Figure 9.1(b). The $O(n) \times O(n^2)$ area bound is due to the fact that the sum of the denominators of the slopes assigned to the edges, that corresponds to the maximum height of a drawing constructed with this algorithm, is $\sum_{i=1}^{n-1} 1 = n - 1$, while the sum of the numerators, that corresponds to the maximum width, is $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$.

In [ACB$^+$12] it has also been proved that in any straight-line monotone drawing of a tree, the length of each edge can be arbitrarily modified without affecting planarity and monotonicity. This is formalized in the following property.



(a)



(b)

Figure 9.1: (a) A drawing Γ of an embedded planar graph G_ϕ. (b) An upright spanning tree of G_ϕ. (c) A spanning tree of G_ϕ that is not upright.

Property 9.1 [ACB$^+$12] *Let Γ be a straight-line monotone drawing of a tree T. Then, any drawing Γ' of T such that the slopes of each edge $e \in T$ in Γ' is the same as the slope of e in Γ is monotone. Also, the slopes of any two leaf-edges e' and e'' of T in Γ are such that e' and e'' diverge, which implies that the elongations of e' and e'' do not cross each other.*

9.3 Poly-line Monotone Drawings of Embedded Planar Graphs

In this section we study monotone drawings of embedded planar graphs. We remark that it is still unknown whether every planar graph admits a straight-line monotone drawing in the variable embedding setting, while it is known that straight-line monotone drawings do not always exist if the embedding of the graph is fixed [ACB$^+$12]. We therefore investigate monotone drawings with bends along some edges, and we show that two bends per edge are always sufficient and sometimes necessary for the existence of a monotone drawing in the fixed embedding setting.

We need some preliminary definitions. An *upright spanning tree* T of an embedded planar graph G_ϕ is a rooted ordered spanning tree of G_ϕ such that:

(i) T preserves the planar embedding of G_ϕ;

(ii) the root of T is a vertex r of the outer face of G_ϕ;

(iii) there exists a planar drawing of G_ϕ that contains an upward drawing of T such that no edge of $G_\phi \setminus T$ passes below r.

Figures 9.2(b) and 9.2(c) show two different ordered spanning trees of the embedded planar graph G_ϕ of Figure 9.2(a) rooted at node 1. The first tree is an upright spanning tree, while the second one is not. Namely, even if the tree in Figure 9.2(c) preserves the embedding ϕ, edge $(7, 3)$ passes below vertex 1.

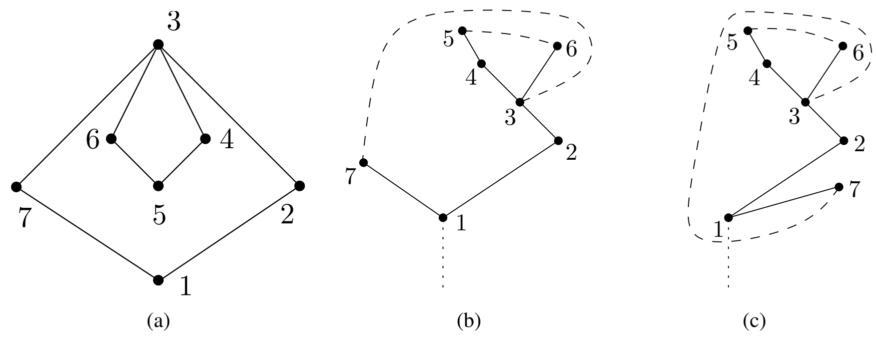


Figure 9.2: (a) A drawing Γ of an embedded planar graph G_ϕ. (b) An upright spanning tree of G_ϕ. (c) A spanning tree of G_ϕ that is not upright.

Given an embedded planar graph G_ϕ, an upright spanning tree T of G_ϕ can be computed as follows. Construct any planar straight-line drawing Γ of G_ϕ. Orient the edges of G_ϕ in Γ according to the upward direction, by giving a random orientation to horizontal edges. Let r be a source on the outer face of G_ϕ with the smallest y-coordinate in Γ. Then, compute any spanning tree T of G_ϕ rooted at r such that the left-to-right order of the children of r in T is consistent with the left-to-right order of the neighbors of r in Γ and the left-to-right order of the children of each vertex w in T is consistent with the clockwise order of the neighbors of w in G_ϕ, computed starting from the edge connecting w to its parent in T.

Let T be an upright spanning tree of G_ϕ. The *rgbb-coloring $C(G_\phi, T)$ of G_ϕ with respect to T* is a coloring of the edges of G_ϕ with four colors such that:

- An edge is colored *black* if it belongs to T;
- an edge is colored *green* if it connects two leaves of T;
- an edge is colored *red* if it connects a leaf to an internal vertex of T;
- an edge is colored *blue* if it connects two internal vertices of T.

We now describe an algorithm that, given an embedded planar graph G_ϕ with n vertices, an upright spanning tree T of G_ϕ, and the rgbb-coloring $C(G_\phi, T)$ of G_ϕ with respect to T, constructs a monotone drawing Γ of G_ϕ such that each black or green edge is drawn as a straight-line segment, each red edge has one bend, and each blue edge has two bends. We call this algorithm MONOTONE-FIXED-EMBEDDING; Lemma 9.1 will prove that it correctly computes a monotone drawing with curve complexity 2 and $O(n^3)$ area, in $O(n)$ time.

First, starting from G_ϕ and T, algorithm MONOTONE-FIXED-EMBEDDING constructs a graph G'_ϕ and an upright spanning tree T' of G'_ϕ such that:

(i) G'_ϕ is a 2-subdivision of G_ϕ;

(ii) T is a subtree of T';

(iii) all the edges of G'_ϕ that are not in T' connect two leaves of T'.

Graphs G'_ϕ and T' are constructed as follows. Initialize $G'_\phi = G_\phi$ and $T' = T$. Subdivide each red edge (s, t) of G'_ϕ with a dummy vertex k and add edge (t, k) to T', where t is the internal vertex of T'. Subdivide each blue edge (s, t) of G'_ϕ twice with two dummy vertices k and z, and add edges (s, k) and (t, z) to T'. Figures 9.3(a) and 9.3(b) show a graph G_ϕ with an upright spanning tree T and the corresponding graph G'_ϕ with its upright spanning tree T' satisfying (i)–(iii).

Then, a monotone drawing of G_ϕ with curve complexity 2 is constructed by first computing a straight-line monotone drawing of G'_ϕ and then replacing each subdivision vertex with a bend; see Figure 9.3(c).

The straight-line monotone drawing of G'_ϕ is computed in two steps. First, with *Algorithm DFS-based* [ACB+12], construct a straight-line monotone drawing of T'. Second, add the remaining (non-tree) edges as straight-line segments by suitably elongating the leaf-edges of T', as described in the following. Observe that, this results

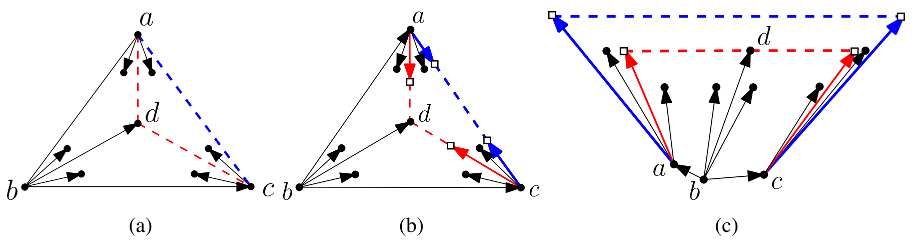


Figure 9.3: (a) A graph G_ϕ with an upright spanning tree T rooted at vertex b. Solid edges belong to T, while dashed edges do not. Blue edges are thicker than red edges, which are thicker than black edges. (b) The corresponding graph G'_ϕ with its upright spanning tree T' constructed by algorithm MONOTONE-FIXED-EMBEDDING. Solid edges belong to T', while dashed edges do not. Subdivision dummy vertices are drawn as squares. (c) A straight-line monotone drawing of G'_ϕ that corresponds to a monotone drawing of G_ϕ with bent edges.

in using two segments for red edges and three segments for blue edges when dummy vertices are replaced by bends.

Consider any leaf-edge (u, v), where v is the leaf of T'. Observe that, as *Algorithm DFS-based* assigns slopes $\frac{1}{1}, \frac{2}{1}, \dots, \frac{n-1}{1}$ to the edges of T', the elongation of (u, v) intersects at an integer grid point each vertical line $x = k$, where k is any integer value greater than the x-coordinate of u. Moreover, as the leaf-edge elongations do not cross, such intersections appear in the same order on each vertical line $x = k'$, where k' is any integer value greater than the x-coordinate of every internal vertex of T'; see Figure 9.4(a).

Another key observation is that the graph induced by the leaves of T' is outerplanar and can be augmented, by adding dummy edges, to a biconnected outerplanar graph G_L such that every internal face of G_L is a 3-cycle and the order of the vertices on the outer face of G_L is the same as the left-to-right order of the leaves of T'; see Figure 9.4(b).

The vertices of G_L are assigned to levels in such a way that the end-vertices of each edge of G_L are either on the same level or on adjacent levels, as follows. The first and the last vertex in the left-to-right order of the leaves of T' have level 1. Note that these two vertices are adjacent, as G_L is a biconnected outerplanar graph and the order of the vertices on its outer face is the same as the left-to-right order of the leaves of T'. Then, starting from this edge, consider any edge (u, v) on the outer face of the graph induced by the vertices whose level has been already assigned. Consider the unique vertex w that is connected to both u and v, and whose level has not been assigned yet, if any. Note that, either u and v have the same level i or one of them has level i and the other has level $i + 1$. In both cases, assign level $i + 1$ to w, as shown in Figures 9.4(b) and 9.4(c). Let l be the number of levels of G_L. The drawing of G'_ϕ is completed by placing all the vertices at level i, with $i = 1, \ldots, l$, on a vertical line $x = k + l - i + 1$, where k is the x-coordinate of the rightmost internal vertex of T'; see Figure 9.4(c).

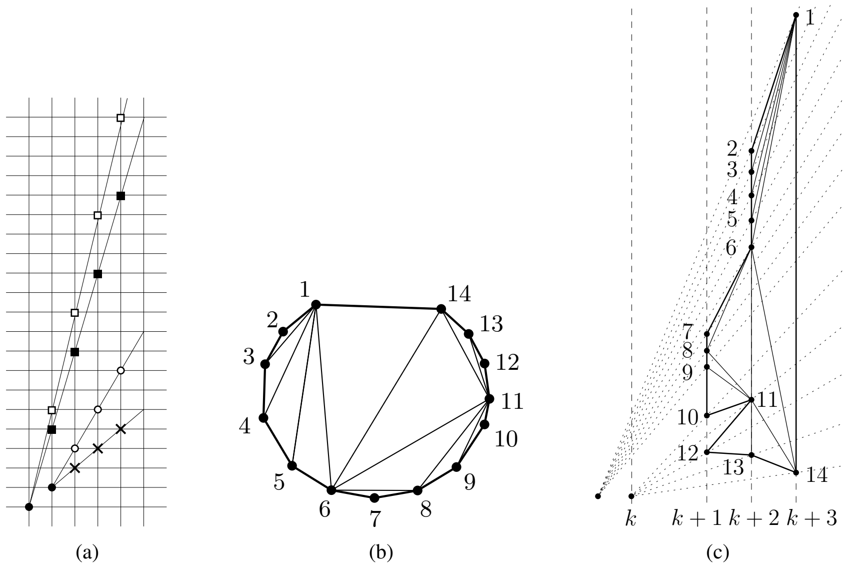


Figure 9.4: (a) Leaf-edge elongations have integer intersections with all the vertical lines in the same order. (b) An augmented graph G_L. (c) The drawing of G_L, where the number l of levels is equal to 3.

Lemma 9.1 *Given an embedded planar graph G_ϕ with n vertices, an upright spanning tree T of G_ϕ, and the rgbb-coloring $C(G_\phi, T)$ of G_ϕ with respect to T, algorithm* MONOTONE-FIXED-EMBEDDING *constructs a planar monotone drawing Γ of G_ϕ with curve complexity 2 and $O(n) \times O(n^2)$ area in $O(n)$ time.*

Proof: First, we prove that the drawing of the auxiliary graph G'_ϕ that is a subdivision of G_ϕ is planar and monotone. Namely, by Property 9.1, the slopes assigned to the edges of the spanning tree T' of G'_ϕ by *Algorithm DFS-based* [ACB+12] are such that for any elongation of the edges of T', the resulting drawing is planar and monotone. Further, since T' is an upright spanning tree of G'_ϕ, it preserves the planar embedding of G'_ϕ and there are no edges going below the root. Also, recall that algorithm MONOTONEFIXEDEMBEDDING places all the vertices at level i, with $i = 1, \ldots, l$, on a vertical line $x = k + l - i + 1$, where G_L is the biconnected outerplanar graph composed of the edges of $G'_\phi \setminus T'$ plus the dummy edges needed to make it biconnected, l is the number of levels of G_L, and k is the x-coordinate of the rightmost internal vertex of T'. This placement, together with the fact that each such vertical line intersects the elongations of all the leaf-edges in the same order, ensures the planarity of the straight-line drawing of G_L. Further, as the order of the vertices on the outer face of G_L is the same as the left-to-right order of the leaves of T', the edges of T' do not cross any edge of G_L. Hence the drawing of G'_ϕ is planar. Finally, since T' is a spanning tree of G'_ϕ, any two vertices of G'_ϕ are connected by a monotone path composed only of edges of T', and hence the drawing of G'_ϕ is also monotone.

The planarity of the drawing Γ of G_ϕ comes from the fact that Γ is obtained by just replacing the dummy vertices in the drawing of G'_ϕ with bends. To see that Γ is monotone, observe that any monotone path in the drawing of G'_ϕ traversing a leaf-edge of T' has the corresponding leaf as an end-vertex, and if such a leaf is a subdivision dummy vertex of any non-black edge, then it does not belong to G_ϕ. Hence, all the monotone paths in G_ϕ are composed only of edges of T, whose drawing is monotone since it is a subtree of T'. Also, Γ is obtained by replacing dummy vertices with bends in the drawing of G'_ϕ, which is a straight-line drawing; since every edge is subdivided at most twice (namely, each red edge is subdivided once and each blue edge is subdivided twice), drawing Γ has curve complexity 2.

The area of Γ can be derived from that required by the drawing of G'_ϕ. Namely, the elongation of the leaf-edges of T', computed by algorithm MONOTONEFIXEDEMBEDDING in order to reinsert the edges of $G'_\phi \setminus T'$ as straight-line segments, does not asymptotically increase the $O(n) \times O(n^2)$ area of the drawing of T' obtained with *Algorithm DFS-based*. Indeed, since the number of vertical lines added to host the drawing of G_L equals the number l of levels assigned to the vertices of G_L, and since l is bounded by the number of leaves (which is $O(n)$), the area of Γ remains $O(n) \times O(n^2)$.

Concerning the time complexity of the algorithm, we analyze the time required by each individual step. The computation of the graphs T, G'_ϕ, and T' can be easily performed in $O(n)$ time. Also, the slopes of the edges of T' can be computed in linear time with *Algorithm DFS-based* [ACB+12]. The biconnected planar graph G_L can be constructed in $O(n)$ time by augmenting the outerplanar graph induced by the leaves of T' [Cha91]. Finally, the assignment of levels to the vertices of G_L is performed in $O(n)$ time, as each vertex is considered just once and its level is assigned only based on the levels of its two neighbors. Hence, algorithm MONOTONEFIXEDEMBEDDING runs in linear time. □

Note that, by Lemma 9.1, there always exists a monotone drawing Γ of G_ϕ with curve complexity 2 and at most $4n - 10$ bends in total. Namely, since G_ϕ has at most $3n - 6$ edges and every spanning tree of G_ϕ has $n - 1$ edges, and since the edges of the spanning tree are drawn as straight-line segments, drawing Γ has at most $2(3n - 6 - n + 1) = 4n - 10$ bends in total.

In the following we prove that our bound on the maximum number of bends per edge is tight, i.e., we show infinitely many embedded planar graphs that do not admit any embedding-preserving monotone drawing with at most one bend per edge. More interestingly, we prove that every monotone drawing of these graphs contains $\Omega(n)$ edges with two bends, hence its total number of bends is linear in the number of vertices. This implies that in general it is not possible to improve the drawing algorithm of Lemma 9.1 to achieve a sublinear number of bends in total; therefore, the $4n - 10$ bound is asymptotically optimal. We first prove in Lemma 9.2 that there exist embedded planar graphs requiring at least one bend on some edges. Then, based on this lemma, we prove in Lemma 9.3 that there exist infinitely many embedded planar graphs whose monotone drawings require two bends on a linear number of edges.

We first introduce a definition that will be useful to prove the claimed lemmata. The *turn angle* from the edge (u, v) to the edge (v, w) is the smallest angle between the half-line from u through v and the edge (v, w); see Figure 9.5. The turn angle is *positive* if the rotation defined by this angle is clockwise and *negative* if it is counterclockwise.

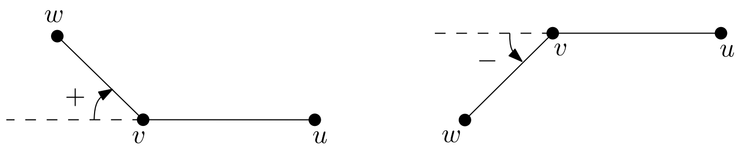


Figure 9.5: A positive and a negative turn angle from edge (u, v) to edge (v, w).

Lemma 9.2 *For every $n \geq 3$ there exists an embedded planar graph G_ϕ with $3n$ vertices and $3n$ edges that does not admit any straight-line monotone drawing.*

Proof: Graph G_ϕ consists of a simple cycle $C = v_1, \dots, v_{2n}$ of length $2n$ and of n degree-1 vertices $u_1, u_3, \dots, u_{2n-1}$, called *legs*, adjacent to vertices $v_1, v_3, \dots, v_{2n-1}$

of C with odd indices, respectively. The embedding of G_ϕ is such that all the legs are inside C, that is, they are incident to the unique internal face of C; see Figure 9.6(a).

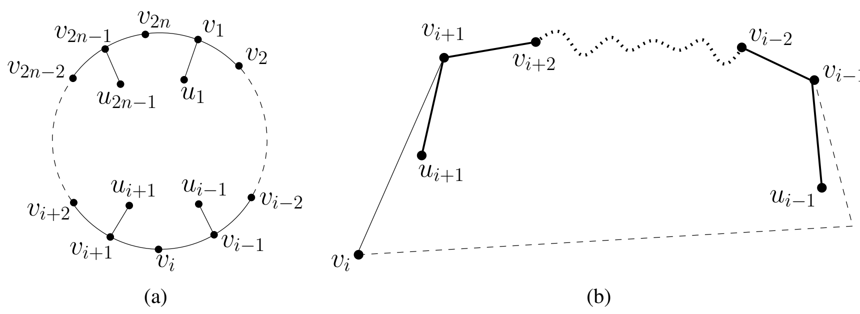


Figure 9.6: (a) A graph G_ϕ with $3n$ vertices that does not admit any embedding-preserving straight-line monotone drawing. (b) Path P_i^2 cannot be monotone.

We prove that there exists no straight-line monotone drawing of G_ϕ. Assume, for a contradiction, that such a straight-line monotone drawing exists. Consider two arbitrary consecutive legs u_{i-1} and u_{i+1} of G_ϕ. Note that there exist only two paths $P_i^1 = u_{i-1}, v_{i-1}, v_i, v_{i+1}, u_{i+1}$ and $P_i^2 = u_{i-1}, v_{i-1}, v_{i-2}, \dots, v_1, v_{2n}, \dots, v_{i+2}, v_{i+1}, u_{i+1}$ connecting u_{i-1} and u_{i+1}.

We show that path P_i^2 cannot be monotone. Refer to Figure 9.6(b). Namely, if P_i^2 is monotone then, by Property 9.1, edges (v_{i-1}, u_{i-1}) and (v_{i+1}, u_{i+1}) diverge, as u_{i-1} and u_{i+1} are leaves of P_i^2. Hence, in order to connect v_{i-1} to v_{i+1} with a polyline while keeping u_{i-1} and u_{i+1} inside the polygon representing C, at least three straight-line segments are necessary. However, only two edges, namely (v_{i-1}, v_i) and (v_i, v_{i+1}), lie between v_{i-1} and v_{i+1} in $C \setminus P_i^2$, a contradiction.

Thus, for any pair of consecutive legs u_{i-1} and u_{i+1} the monotonicity between them is provided by path P_i^1. We show that this leads to a contradiction.

Let α_i, $i = 1, \dots, 2n$, be the turn angle from edge (v_{i-1}, v_i) to edge (v_i, v_{i+1}), where the indices are taken modulo $2n$. Let also β_i and γ_i, $i = 1, 3, \dots, 2n-1$ be the turn angles from (v_{i-1}, v_i) to (v_i, u_i), and from (u_i, v_i) to (v_i, v_{i+1}), respectively. Note that:

$$\sum_{i=1}^{2n} \alpha_i = 2\pi. \tag{9.1}$$

Also, as shown in Figs. 9.7(a-d), for each $i = 1, 3, \dots, 2n-1$, regardless of whether the internal angle at v_i is convex or reflex and of the position of the leg u_i, the following equation holds:

$$\beta_i + \gamma_i = \alpha_i + \pi \tag{9.2}$$

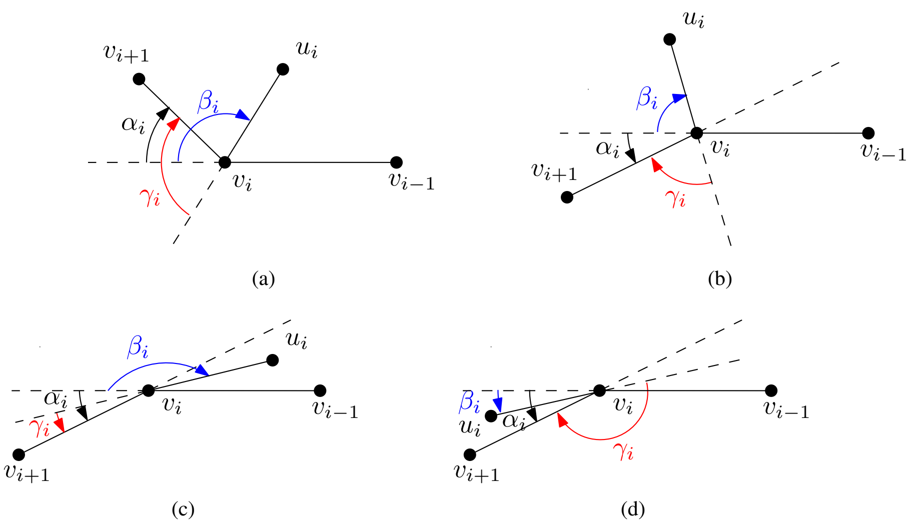


Figure 9.7: The proof that equality $\beta_i+\gamma_i = \alpha_i+180$ holds for each $i = 1, 3, \dots, 2n-1$. (a) The internal angle at v_i is convex. Equality $|\beta| + |\gamma| = 180 + |\alpha|$ holds and $\alpha, \beta, \gamma \geq 0$. (b) The internal angle at v_i is reflex and the leg u_i is between the half-line from v_{i-1} through v_i and the half-line from v_{i+1} through v_i in the circular ordering around v_i. Equality $|\beta| + |\gamma| = 180 - |\alpha|$ holds, and $\alpha \leq 0$; $\beta, \gamma \geq 0$. (c) The internal angle at v_i is reflex and the leg u_i is between the half-line from v_{i+1} through v_i and edge (v_{i-1}, v_i) in the circular ordering around v_i. Equality $|\beta| - |\gamma| = 180 - |\alpha|$ holds, and $\alpha, \gamma \leq 0$; $\beta \geq 0$. (d) The internal angle at v_i is reflex and the leg u_i is between the half-line from v_{i-1} through v_i and edge (v_{i+1}, v_i) in the circular ordering around v_i. Equality $|\gamma| - |\beta| = 180 - |\alpha|$ holds, and $\alpha, \beta \leq 0$; $\gamma \geq 0$.

Further, as path P_i^1 from u_{i-1} to u_{i+1} through v_i is monotone, we have that, for each $i = 1, 3, \dots, 2n-1$, the following inequality holds; see Figure 9.8.

$$\gamma_{i-1} + \alpha_i + \beta_{i+1} < \pi \tag{9.3}$$

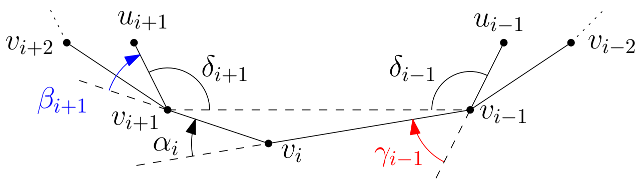


Figure 9.8: $\delta_{i-1} + \delta_{i+1} + \gamma_{i-1} + \alpha_i + \beta_{i+1} = 2\pi$. As legs u_{i-1} and u_{i+1} diverge, $\delta_{i-1} + \delta_{i+1} \geq \pi$. Hence, $\gamma_{i-1} + \alpha_i + \beta_{i+1} < \pi$

By summing up inequality 9.3 over $i = 1, 3, \ldots, 2n-1$, we get:

$$\begin{aligned} & \gamma_1 + \alpha_2 + \beta_3 + \gamma_3 + \alpha_4 + \beta_5 + \cdots + \gamma_{2n-1} + \alpha_n + \beta_1 \\ = \; & (\beta_1 + \gamma_1) + \alpha_2 + (\beta_3 + \gamma_3) + \alpha_4 + \cdots + (\beta_{2n-1} + \gamma_{2n-1}) + \alpha_n \\ < \; & n\pi \end{aligned}$$

Applying equation 9.2, we get $(\alpha_1+\pi)+\alpha_2+(\alpha_3+\pi)+\alpha_4+\cdots+(\alpha_{2n-1}+\pi)+\alpha_n = \sum_{i=1}^{2n} \alpha_i + n\pi < n\pi$. By equation 9.1, we get $(n+2)\pi < n\pi$, a contradiction. □

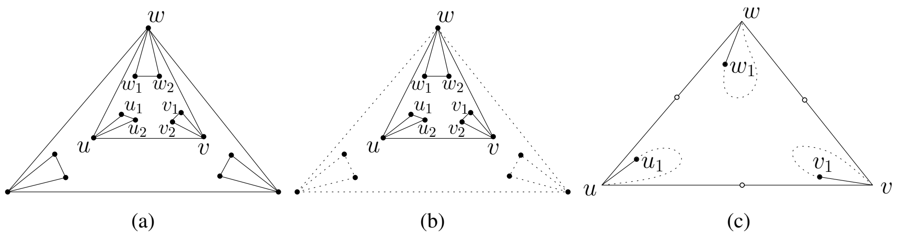


Figure 9.9: (a) A graph G_ϕ with $n = 15$ vertices, which coincides with a graph G_ϕ^3 constructed from G_ϕ^2 by adding vertices $u_1, u_2, v_1, v_2, w_1, w_2$ inside the triangular face u, v, w. (b) A subgraph G_ϕ^t of G_ϕ induced by a triangle (u, v, w) and all the vertices inside it. (c) A subdivision (white circles) of the subgraph G_ϕ^h (solid edges) of G_ϕ^t induced by u, v, w, u_1, v_1, w_1. By Lemma 9.2, it does not admit a straight-line monotone drawing.

Lemma 9.3 *For every odd $n \geq 9$ there exists an embedded planar graph G_ϕ with n vertices and $\frac{3}{2}(n-1)$ edges such that every monotone drawing of G_ϕ has at least $\frac{n-3}{6}$ edges with at least two bends and thus at least $\frac{n-3}{3}$ bends in total.*

Proof: Consider an odd integer $n \geq 9$. We construct G_ϕ iteratively. Let G^1_ϕ be a triangle graph. Graph G^i_ϕ is constructed from G^{i-1}_ϕ as follows. Initialize $G^i_\phi = G^{i-1}_\phi$. Let (u, v, w) be a triangular internal face of G^i_ϕ. Add 6 new vertices $u_1, u_2, v_1, v_2, w_1, w_2$ and 9 new edges (u, u_1), (u, u_2), (u_1, u_2), (v, v_1), (v, v_2), (v_1, v_2), (w, w_1), (w, w_2), (w_1, w_2) to G^i_ϕ in such a way that all the new vertices are inside (u, v, w). Note that the n-vertex graph G^i_ϕ is planar and has $\frac{3}{2}(n-1)$ edges; see Figure 9.9(a).

We prove that any monotone drawing of G_ϕ has at least $\frac{n-3}{6}$ edges with at least two bends. Let G^t_ϕ be a subgraph of G_ϕ induced by a triangle (u, v, z) of G_ϕ and by all the vertices drawn inside it (see Figure 9.9(b)). Let Γ be a drawing of G_ϕ and let Γ^t be a drawing of G^t_ϕ that coincides with Γ restricted to the edges of G^t_ϕ. We have the following.

Claim 9.1 Γ *is a monotone drawing only if* Γ^t *is a monotone drawing.*

Proof: If Γ^t is not monotone there are two vertices a and b of G^t_ϕ that are not connected by any monotone path in Γ^t. Assume for contradiction that Γ is monotone. Then, a and b are connected by a monotone path P in Γ. As P does not lie entirely in G^t_ϕ, it passes twice through a cut vertex c which is a common vertex of G^t_ϕ and G_ϕ. Thus, the path obtained from P by removing the part between the two occurrences of c is a monotone path between a and b only composed of edges of G^t_ϕ, a contradiction. □

Claim 9.2 *In any monotone drawing of G^t_ϕ, at least one of the edges (u, v), (v, w), (w, u) has two bends.*

Proof: Consider the subgraph G^h_ϕ of G^t_ϕ induced by vertices u, v, w, u_1, v_1, and w_1; see Figure 9.9(c). Every monotone drawing of G^t_ϕ restricted to the edges of G^h_ϕ is a monotone drawing of G^h_ϕ. Thus, if there exists a monotone drawing of G^t_ϕ such that none of the edges (u, v), (v, w), (w, u) has two bends, then G^h_ϕ also has such a monotone drawing. However, we show that G^h_ϕ does not admit any such drawing. Let G^s_ϕ be a 1-subdivision of G^h_ϕ. Note that, G^s_ϕ satisfies the preconditions of Lemma 9.2, that is, it is composed of a cycle plus a set of internal legs connected to every second vertex of the cycle, and hence it does not admit a straight-line monotone drawing. Thus, G^h_ϕ does not admit a monotone drawing with curve complexity 1, as bends on the edges of G^h_ϕ correspond to the subdivision vertices in G^s_ϕ. □

Claim 9.1 implies that the drawing of every subgraph defined as G^t_ϕ is monotone, and Claim 9.2 implies that in any monotone drawing of such a subgraph one of the edges of the outer triangle of G^t_ϕ has two bends. As the number of different triangles that contain a subgraph G^t_ϕ in their interior is $\frac{n-3}{6}$, any monotone drawing of G_ϕ has at least $\frac{n-3}{6}$ edges with two bends, and thus $\frac{n-3}{3}$ bends in total. □

Lemmata 9.1 and 9.3 together provide a tight bound on the curve complexity of monotone drawings in the fixed embedding setting. The following theorem summarizes the main contribution of this section.

Theorem 9.1 *Every embedded planar graph with n vertices admits a monotone drawing with curve complexity 2, at most $4n - 10$ bends in total, and $O(n) \times O(n^2)$ area; such a drawing can be computed in $O(n)$ time. Also, there exist infinitely many embedded planar graphs any monotone drawing of which requires two bends on $\Omega(n)$ edges.*

9.4 Straight-line Monotone Drawings of Embedded Planar Graphs

In this section we prove that there exist meaningful subfamilies of embedded planar graphs that can be realized as straight-line monotone drawings. In particular, we prove that the class of outerplane graphs and the class of embedded planar biconnected graphs have this property.

9.4.1 Outerplane Graphs

An embedded planar graph G_ϕ is an *outerplane graph* if all its vertices are incident to the outer face. We prove the following result.

Theorem 9.2 *Every n-vertex outerplane graph admits a straight-line monotone drawing, which can be computed in $O(n)$ time and requires $O(n) \times O(n^2)$ area.*

Proof: Let T be an upright spanning tree of G_ϕ obtained by performing a "rightmost DFS" visit of G_ϕ, that is, a DFS visit in which the neighbors of each vertex are considered in counterclockwise order; see Figure 9.10(a). Consider a decomposition of G_ϕ into its maximal biconnected components. Observe that, for each maximal biconnected component B that is connected to the root of T through a cut-vertex v, T contains all the edges of B except for its internal chords (dashed edges in Figure 9.10(a)) and for its leftmost edge incident to v (dotted edges in Figure 9.10(a)).

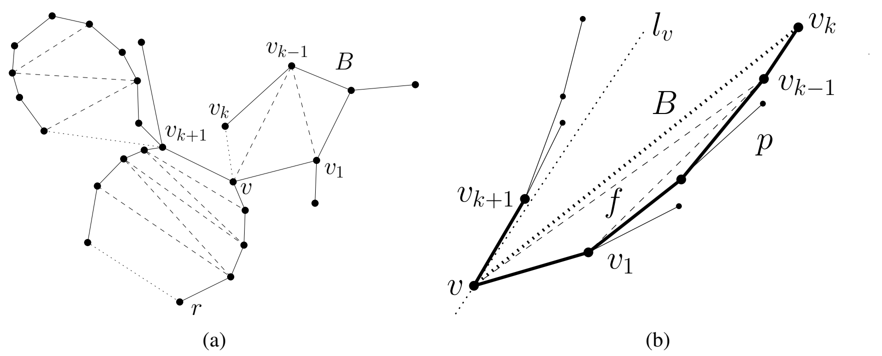


Figure 9.10: (a) An outerplane graph G_ϕ and its upright spanning tree T obtained by performing a "rightmost DFS" visit. Edges of T are solid segments. (b) A strictly convex drawing of a maximal biconnected component B of G_ϕ.

A straight-line monotone drawing of G_ϕ is constructed by first computing a straight-line monotone drawing of T, with *Algorithm DFS-based* [ACB$^+$12], and then reinserting the edges not in T as straight-line segments.

In order to prove that such edges can be reinserted as straight-line segments without creating crossings, for each maximal biconnected component B consider the path $p = (v, v_1, \ldots, v_k)$ that is composed of the edges belonging both to B and to T. According to *Algorithm DFS-based* [ACB$^+$12], the slopes of the edges of p are all positive and are increasing with respect to the distance from v in p. Hence, path p is drawn in T as a polygonal line "convex on the left side", that is, the straight-line segment connecting any two non-consecutive vertices of p completely lies to the left of p; see Figure 9.10(b). Thus, when reinserting edge (v, v_k) as the straight-line segment between v and v_k, the boundary of B, namely the cycle composed of the edges of p plus (v, v_k), delimits a strictly-convex region f.

We show that f does not contain any other vertex of T. In particular, it is sufficient to show that the vertex v_{k+1} such that edge (v, v_{k+1}) follows (v, v_1) in the counterclockwise order of the edges around v in T lies outside f.

First, consider a straight line l_v through v that is parallel to edge (v_{k-1}, v_k). According to *Algorithm DFS-based*, the slope of (v_{k-1}, v_k) is the greatest among the slopes of the edges of p. Hence, vertex v_k is to the right of l_v. Thus, the slope of (v_{k-1}, v_k) is greater than or equal to the slope of (v, v_k), with the equality being possible only if $v = v_{k-1}$. Since the slope of (v, v_{k+1}) is greater than the slope of (v_{k-1}, v_k), it follows that v_{k+1} lies outside f; see Figure 9.10(b).

From the discussion above, it follows that f is an empty strictly-convex region, and the chords of B can be reinserted as straight-line segments while maintaining planarity.

The obtained drawing is monotone since every pair of vertices is connected by a monotone path composed only of edges of T.

The area of the drawing is the same as the area of T computed by *Algorithm DFS-based*, namely $O(n) \times O(n^2)$. The drawing can be computed in $O(n)$ time. Namely, drawing T by using *Algorithm DFS-based* takes $O(n)$ time [ACB$^+$12], and the same holds for reinserting missing edges. □

9.4.2 Biconnected Graphs

It is known [ACB$^+$12] that straight-line monotone drawings of biconnected planar graphs in the variable embedding setting can always be computed by means of an algorithm that, for each component μ of the SPQR-tree of the input graph, draws the pertinent graph of μ inside a shape, called *boomerang*, while respecting some geometrical properties concerning planarity and monotonicity. In Figure 9.11(a) the drawing of a parallel component inside a boomerang is depicted.

Note that, this algorithm preserves any given embedding of the input graph, except for the case in which the graph contains a parallel component μ_P whose poles are connected by an edge. In fact, in this case, such an edge is always drawn either as the first or as the last element in the order of the children of μ_P, as in Figure 9.11(a), while this might not happen in the given embedding.

On the other hand, when this type of edge exists, an embedding-preserving monotone drawing could be obtained by adding one bend along each such edge in order to place it in its correct position, as in Figure 9.11(b), hence obtaining a monotone drawing with curve complexity 1.

In this section we prove that in fact it is possible to compute an embedding-preserving monotone drawing of every embedded biconnected planar graph with no bends at all. Intuitively, when dealing with a parallel component μ_P, we use a shape called *diamond* instead of a boomerang to draw the pertinent graph of μ_P. A diamond is composed of two mirrored copies of a boomerang that are separated by the line through the poles. Then, the edge between the poles, if it exists, is drawn as a straight-line segment, the components preceding it in the order of the children of μ_P are drawn in one of the copies, and the components following it are drawn in the other copy. See Figure 9.11(c).

In the following we first give a more detailed description of the algorithm for the variable embedding setting [ACB$^+$12] and then we describe our algorithm, that

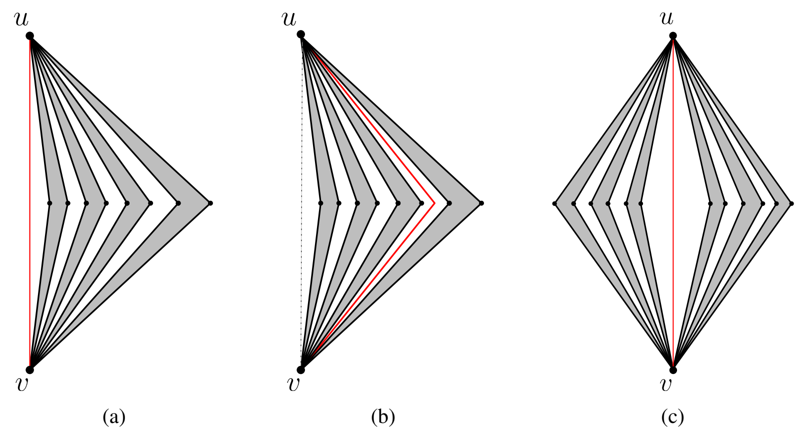


Figure 9.11: (a) The algorithm in [ACB$^+$12] places an edge between the poles of a parallel component μ_P either as the first or as the last element in the order of the children of μ_P. (b) An embedding-preserving monotone drawing of μ_P with curve complexity 1. (c) An embedding-preserving straight-line monotone drawing of μ_P produced with the algorithm described in this section.

we call BICOFIXEDEMBEDDING, to compute an embedding-preserving straight-line monotone drawing of every embedded biconnected planar graph G.

We start by giving some definitions. A boomerang *boom*(μ) is a quadrilateral[2] $(N_\mu, E_\mu, S_\mu, W_\mu)$ such that W_μ is inside triangle $\triangle(N_\mu, S_\mu, E_\mu)$ and $2\alpha_\mu + \beta_\mu < \frac{\pi}{2}$, where $\alpha_\mu = \widehat{W_\mu S_\mu E_\mu} = \widehat{W_\mu N_\mu E_\mu}$ and $\beta_\mu = \widehat{W_\mu S_\mu N_\mu} = \widehat{W_\mu N_\mu S_\mu}$; see Figure 9.12(a).

A path monotone with respect to a direction d is (α, d)-*monotone* (with $\alpha < \frac{\pi}{2}$) if for each edge e it holds that $d - \alpha < sl(e) < d + \alpha$. A path from a vertex u to a vertex v is an (α, d_1, d_2)-*path* if it is a composition of an (α, d_1)-monotone path from u to a vertex w and of an (α, d_2)-monotone path from w to v.

The algorithm described in [ACB$^+$12] inductively constructs a drawing of any biconnected planar graph G by means of a bottom-up visit of the SPQR-tree of G, as follows. When a component μ with child components $\mu_1, \dots, \mu_k$ is visited, a drawing Γ_μ of the pertinent graph of μ satisfying the properties (A), (B), (C) hereunder is constructed by composing the drawings of $\mu_1, \dots, \mu_k$ which, by induction, satisfy the

[2]In [ACB$^+$12] points $N_\mu, E_\mu, S_\mu, W_\mu$ are denoted as $p_N(\mu), p_E(\mu), p_S(\mu), p_W(\mu)$, respectively.

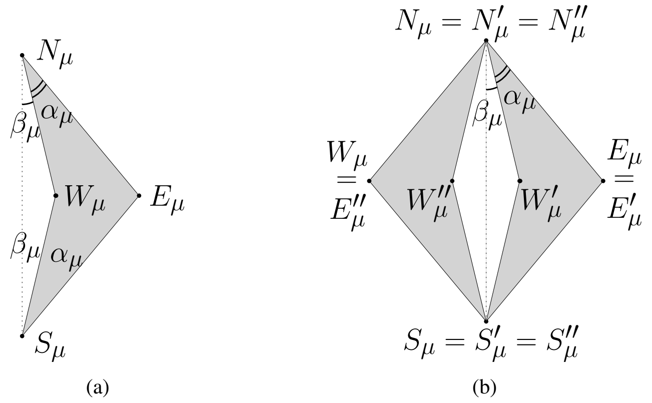


Figure 9.12: (a) A boomerang. (b) A diamond.

same properties. The composition is based on whether μ is an S-, a P-, a Q-, or an R-node.

(A) Γ_μ is monotone;

(B) Γ_μ is planar and, with the possible exception of edge (u, v), it is contained inside $boom(\mu)$, with u drawn on N_μ and v on S_μ;

(C) each vertex $w \in pert(\mu)$ belongs to a $(\alpha_\mu, -d_N(\mu), d_S(\mu))$-path from u to v, where $d_N(\mu)$ (resp., $d_S(\mu)$) is the half-line from E_μ through N_μ (resp., S_μ).

Algorithm BICOFIXEDEMBEDDING for the fixed-embedding case also relies on a bottom-up visit of the SPQR-tree of G. However, in order to cope with the possible existence of edges connecting the poles of some P-nodes, we defined a new shape, called *diamond* and denoted by $diam(\mu)$, as a convex quadrilateral (N_μ, E_μ, S_μ, W_μ) composed of two boomerangs $boom'(\mu) = (N'_\mu, E'_\mu, S'_\mu, W'_\mu)$ and $boom''(\mu) = (N''_\mu, E''_\mu, S''_\mu, W''_\mu)$ such that $N_\mu = N'_\mu = N''_\mu$, $S_\mu = S'_\mu = S''_\mu$, $E_\mu = E'_\mu$, and $W_\mu = E''_\mu$. A diamond $diam(\mu)$ is depicted in Figure 9.12(b).

Then, when considering a P-node μ having an edge e between its poles, one of the two boomerangs composing the diamond contains the child components of μ that come before e in the ordering of the components around the poles, while the other boomerang contains the other components. In this case, the drawing Γ_μ of $pert(\mu)$ must still satisfy Properties (A), (B), and (C), but in Property (B) the whole drawing (including edge (u, v)) is drawn inside the diamond. On the other hand, S- and R-nodes, and P-nodes without an edge between the poles are still drawn inside boomerangs; however, slight modifications are required in their drawing algorithm, as they could now have child P-nodes drawn inside diamonds to handle.

We describe how algorithm BICOFIXEDEMBEDDING computes the drawing Γ_μ for each type of node μ.

Q-node: If μ is a Q-node, $pert(\mu)$ consists only of one edge between the poles. Draw it between points N_μ and S_μ of $boom(\mu)$.

S-node: If μ is an S-node, $pert(\mu)$ must be drawn inside a boomerang $boom(\mu)$. Recall that each child component $\mu_1, \dots, \mu_k$ might be inductively drawn either inside a boomerang or inside a diamond. Apply the same algorithm as for the variable embedding case. Namely, for each child component μ_i, with $i = 1, \dots, k-1$, place points N_{μ_i} and S_{μ_i} on the bisector line of $\widehat{W_\mu N_\mu E_\mu}$, and place N_{μ_k} and S_{μ_k} on the bisector line of $\widehat{W_\mu S_\mu E_\mu}$. Hence, even if μ_i $(1 \leq i \leq k)$ is a parallel node drawn inside a diamond, it is still possible to choose angles α_{μ_i} and β_{μ_i} small enough to fit $diam(\mu_i)$ inside $boom(\mu)$, hence satisfying Property (B); see Figure 9.13(a). Properties (A) and (C) are easily satisfied by induction, as in the variable embedding case.

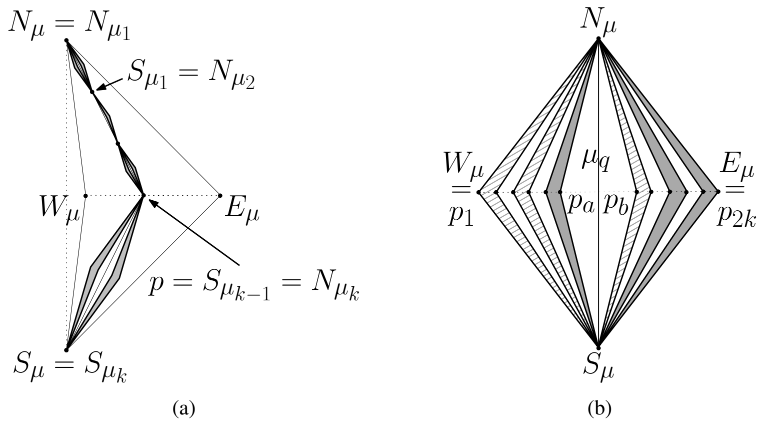


Figure 9.13: Construction of a drawing of $pert(\mu)$ satisfying the inductive hypothesis. (a) μ is an S-node. (b) μ is a P-node. Grey shaded boomerangs contain child components, while grey tiled boomerangs do not.

More formally, let p be the intersection point between segment $\overline{W_\mu E_\mu}$ and the bisector line of $\widehat{W_\mu N_\mu E_\mu}$. Consider k equidistant points $p_1, \ldots, p_k$ on segment $\overline{N_\mu p}$ such that $p_1 = N_\mu$ and $p_k = p$. For each μ_i, with $i = 1, \ldots, k-1$, consider a boomerang $boom(\mu_i) = (N_{\mu_i}, E_{\mu_i}, S_{\mu_i}, W_{\mu_i})$ such that $N_{\mu_i} = p_i$, $S_{\mu_i} = p_{i+1}$, and such that E_{μ_i} and W_{μ_i} determine $\beta_{\mu_i} + 2\alpha_{\mu_i} < \frac{\alpha_\mu}{2}$. Also, consider a boomerang $boom(\mu_k) = (N_{\mu_k}, E_{\mu_k}, S_{\mu_k}, W_{\mu_k})$ such that $N_{\mu_k} = p$, $S_{\mu_k} = S_\mu$, and such that E_{μ_k} and W_{μ_k} determine $\beta_{\mu_k} + 2\alpha_{\mu_k} < \frac{\alpha_\mu}{2}$. Further, for each μ_i ($1 \leq i \leq k$) that is a P-node, consider a diamond $diam(\mu_i)$ composed of two mirroring copies of $boom(\mu_i)$. Then, for each $i = 1, \ldots, k$ apply the inductive algorithm to μ_i and either $boom(\mu_i)$ or $diam(\mu_i)$.

P-node: If μ is a P-node, μ must be drawn inside a diamond $diam(\mu)$, while all its child components μ_i, with $i = 1, \ldots, k$, are inductively drawn inside boomerangs $boom(\mu_i)$, as none of them can be a P-node.

First, note that at most one child component μ_q of μ can be a Q-node representing an edge between the poles of μ. Draw such an edge, if any, as a straight-line segment between N_μ and S_μ. Then, in order to respect the given embedding around the poles of μ, child components $\mu_1, \ldots, \mu_{q-1}$ are drawn inside boomerangs that are contained in the left-hand side of $diam(\mu)$, while the child components $\mu_{q+1}, \ldots, \mu_k$ are drawn inside boomerangs that are contained in the right-hand side of $diam(\mu)$. In order to ensure that the constructed drawings are monotone, we need to fix a total ordering of the components with respect to the angles they form with the line through N_μ and S_μ. This ordering is implicitly guaranteed among components that are on the same side of such a line. In order to obtain it among all the components, we place components $\mu_1, \ldots, \mu_{q-1}$ inside the $q-1$ boomerangs that are the closest to the line through N_μ and S_μ to the left of it, while components $\mu_{q+1}, \ldots, \mu_k$ inside the $k-q$ boomerangs that are the farthest from the line through N_μ and S_μ to the right of it. Refer to Figure 9.13(b).

More formally, consider two internal points p_a and p_b of segment $\overline{W_\mu E_\mu}$ such that p_a is to the left of segment $\overline{N_\mu S_\mu}$ and p_b is to the right of $\overline{N_\mu S_\mu}$.

Further, consider $2(k-1)$ points $p_1, \ldots, p_{2(k-1)}$ on segment $\overline{W_\mu p_a}$ such that $p_1 = W_\mu$, $p_{2(k-1)} = p_a$, and $\widehat{p_i N_\mu p_{i+1}} = \frac{\alpha_\mu}{2(k-1)-1}$, for each $i = 1, \ldots, 2(k-1)-1$. For each μ_i, with $i = 1, \ldots, q-1$, consider a boomerang $boom(\mu_j) = (N_{\mu_j}, E_{\mu_j}, S_{\mu_j}, W_{\mu_j})$, with $j = i + k - q$, such that $N_{\mu_j} = N_\mu$, $S_{\mu_j} = S_\mu$, $E_{\mu_j} = p_{2i-1}$, and $W_{\mu_j} = p_{2i}$. Then, apply the inductive algorithm to μ_i and $boom(\mu_j)$.

Finally, consider $2(k-1)$ points $p'_1, \ldots, p'_{2(k-1)}$ on segment $\overline{p_b E_\mu}$ such that $p'_1 = p_b$, $p'_{2(k-1)} = E_\mu$, and $\widehat{p'_i N_\mu p'_{i+1}} = \frac{\alpha_\mu}{2(k-1)-1}$, for each $i = 1, \ldots, 2(k-1)-1$. For each μ_i, with $i = q+1, \ldots, k$, consider a boomerang $boom(\mu_i) = (N_{\mu_i}, E_{\mu_i}, S_{\mu_i}, W_{\mu_i})$ such that $N_{\mu_i} = N_\mu$, $S_{\mu_i} = S_\mu$, $W_{\mu_i} = p_{2i-1}$, and $E_{\mu_i} = p_{2i}$. Apply the inductive algorithm to μ_i and $boom(\mu_i)$.

Claim 9.3 *If μ is a P-node, the drawing Γ_μ constructed by algorithm* BICOFIXEDEMBEDDING *satisfies Properties (A)–(C).*

Proof: Property (B) is satisfied by construction. Property (C) is satisfied by induction. We prove that Γ_μ satisfies Property (A). Consider any two vertices $w_a, w_b \in pert(\mu)$. If they belong to the same child component, then there exists a monotone path between them by induction. If they belong to different components μ_a and μ_b, consider the $(\alpha_{\mu_a}, -d_N(\mu_a), d_S(\mu_a))$-path $P_a(u, v)$ from u to v through w_a and the $(\alpha_{\mu_b}, d_N(\mu_b), -d_S(\mu_b))$-path $P_b(v, u)$ from v to u through w_b, which exist by induction (Property C). Suppose that μ_a lies inside the left boomerang while μ_b lies inside the right boomerang, the other cases being analogous. Note that, by construction, this implies $\beta_{\mu_a} + 2\beta_{\mu_a} < \beta_{\mu_b}$. Also, suppose that w_b lies on the $(\alpha_{\mu_b}, d_N(\mu_b))$-monotone subpath of $P_b(v, u)$, the other case being analogous. Refer to Figure 9.14(a).

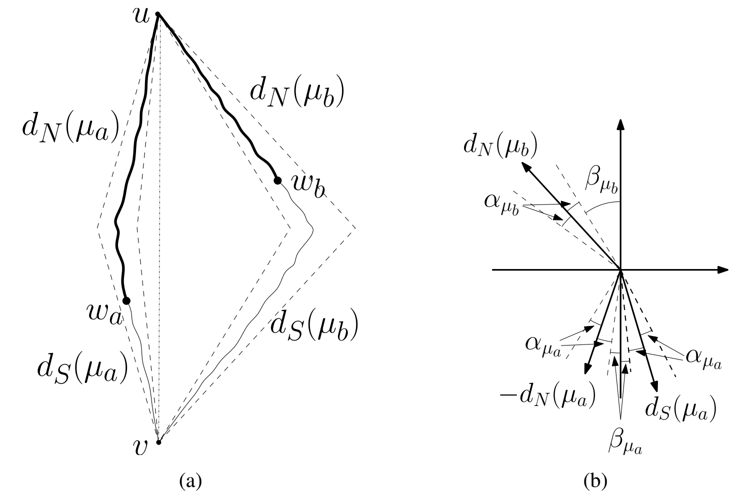


Figure 9.14: If μ is a P-node, the constructed drawing satisfies Property (A), that is, it is monotone. (a) Component μ_a lies inside the left boomerang, μ_b lies inside the right boomerang, and w_b lies on the $(\alpha_{\mu_b}, d_N(\mu_b))$-monotone subpath of $P_b(v, u)$. (b) Since $\beta_{\mu_a} + 2\beta_{\mu_a} < \beta_{\mu_b}$, the directions of all the edges of the path between u and v are inside a wedge whose angle is smaller than π.

Consider the $(\alpha_{\mu_b}, d_N(\mu_b))$-monotone path $P(w_b, u)$ from w_b to u and consider the $(\alpha_{\mu_a}, -d_N(\mu_a), d_S(\mu_a))$-path $P(u, w_a)$ from u to w_a that is a subpath of $P_a(u, v)$. We show that the path $P(w_b, w_a)$ composed of $P(w_b, u)$ and $P(u, w_a)$ is monotone. Refer to Figure 9.14(b). Rotate the coordinate axes in such a way that u and v lie on the y-axis. Then, when translated to the origin of the axes, $d_N(\mu_b)$ is in the second quadrant, $-d_N(\mu_a)$ in the third quadrant, and $d_S(\mu_a)$ in the fourth quadrant. Since $\beta_{\mu_a} + 2\beta_{\mu_a} < \beta_{\mu_b}$, the wedge delimited by $d_N(\mu_b)$ and $d_S(\mu_a)$ and containing the third quadrant has an angle smaller than $\pi - 2\frac{\alpha_\mu}{2k-1}$. Since, by definition, every edge of $P(w_b, u)$ creates an angle with $d_N(\mu_b)$ that is smaller than $\alpha_{\mu_b} = \frac{\alpha_\mu}{2k-1}$ and every edge of $P(u, w_a)$ creates an angle with $d_S(\mu_a)$ that is smaller than $\alpha_{\mu_a} = \frac{\alpha_\mu}{2k-1}$, it follows that the slopes of all the edges of $P(w_b, w_a)$ lie inside a wedge having an angle smaller than π. Hence, $P(w_b, w_a)$ is monotone. $\square$

R-node: If μ is an R-node, $pert(\mu)$ must be drawn inside a boomerang $boom(\mu)$. Also, each child component $\mu_1, \ldots, \mu_k$ might be inductively drawn either inside a boomerang or inside a diamond. As in the S-node case, the drawing algorithm is almost the same as for the variable embedding; see Figure 9.13. We briefly recall this algorithm and highlight the main differences with our variant for the fixed-embedding case. The algorithm consists of two steps. In the first step, a monotone drawing of $skel(\mu)$ is constructed satisfying some properties concerning monotonicity and the slope of the edges, while in the second step angles α_{μ_i} and β_{μ_i} are chosen in order to fit $boom(\mu_i)$ or $diam(\mu_i)$, for each i, inside $boom(\mu)$.

The monotone drawing of $skel(\mu)$ is constructed as follows. First, consider the graph G^* obtained by removing pole v from $skel(\mu)$. Note that, since $skel(\mu)$ is triconnected, G^* admits a convex drawing whose outer face is represented by any strictly convex polygon, as it satisfies all the conditions of Chiba and Nishizeki [CON85, CN88]. Construct a convex drawing Γ^* of G^*, whose outer face is a convex polygon entirely lying in the interior of $boom(\mu)$, except for pole u which is on N_μ, such that the neighbors of v in $skel(\mu)$ are visible from S_μ. See Figure 9.15(a). As Γ^* is convex, it is also monotone [ACM89]. We remark that in [ACB+12] one of the neighbors w of u incident to the outer face of $skel(\mu)$ was placed on point E_μ, while in our algorithm this does not happen. Indeed, placing u on N_μ and w on E_μ makes virtual edge (u, w) be on the boundary of $boom(\mu)$. See Figure 9.15(b). However, this implies that, if the node ν corresponding to (u, w) is a P-node whose pertinent graph has to be drawn inside a diamond $diam(\nu)$, then $diam(\nu)$ cannot be drawn completely inside $boom(\mu)$, hence not satisfying Property (B). See Figure 9.15(c). This problem does not occur in [ACB+12], since $pert(\nu)$ is always drawn inside a boomerang, which can be turned in such a way that it is completely inside $boom(\mu)$.

Second, apply an affine transformation to Γ^*, called *directional-scale* in [ACB+12],

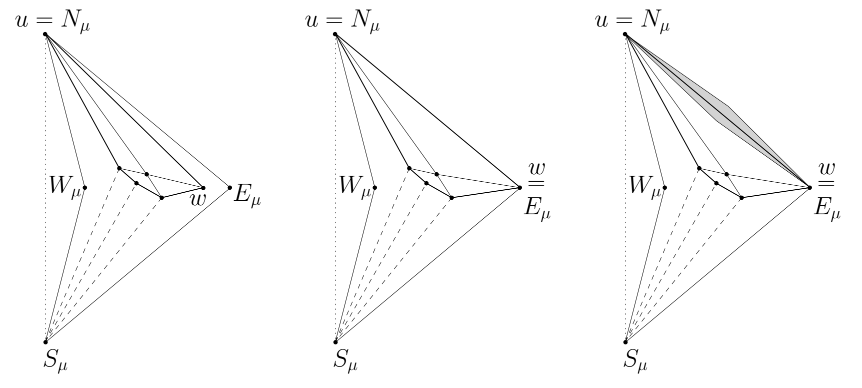


Figure 9.15: (a) A convex polygon entirely lying in the interior of $boom(\mu)$, except for pole u which is on N_μ, such that the neighbors of v in $skel(\mu)$ are visible from S_μ. (b) The corresponding polygon in [ACB$^+$12], in which w is on E_μ. (c) If the node ν corresponding to (u, w) has to be drawn inside a diamond, then it is drawn outside $boom(\mu)$.

in order to make the slopes of all the edges of G^* close enough to $-d_N(\mu)$. See Fig 9.16(a). Namely, the directional-scale applies a scaling to the drawing in a direction that is perpendicular to $-d_N(\mu)$. As proved in [ACB$^+$12] the resulting drawing is still monotone and, for each edge $e \in G^*$, it holds $sl(-d_N(\mu)) - \frac{\alpha_\mu}{2} < sl(e) < sl(-d_N(\mu)) + \frac{\alpha_\mu}{2}$.

Third, construct a drawing $\Gamma(skel(\mu))$ of $skel(\mu)$ by placing v on S_μ in Γ^* and connecting it to its neighbors. Note that v is connected to each vertex w of $skel(\mu)$ by an $(\alpha_\mu, -d_N(\mu), d_S(\mu))$-path that is a composition of the $(\alpha_\mu, d_S(\mu))$-monotone path composed only of edge (v, w'), for some vertex w' adjacent to v, and of the $(\alpha_\mu, -d_N(\mu))$-monotone path connecting w' to w, which exists due to the fact that, for each edge $e \in G^*$, $sl(-d_N(\mu)) - \frac{\alpha_\mu}{2} < sl(e) < sl(-d_N(\mu)) + \frac{\alpha_\mu}{2}$. This, together with the fact that every pair of vertices different from v is connected in $\Gamma(skel(\mu))$ by the same monotone path as in Γ^*, implies that $\Gamma(skel(\mu))$ is monotone.

Finally, consider a drawing $\Gamma'(skel(\mu))$ of a subdivision of $skel(\mu)$ obtained from $\Gamma(skel(\mu))$ by placing the two edges incident to each subdivision vertex on the same straight-line segment. As proved in Lemma 3 of [ACB$^+$12], $\Gamma'(skel(\mu))$ is still a monotone drawing.

Then, in order to obtain Γ_μ, each virtual edge of $skel(\mu)$ is replaced in $\Gamma'(skel(\mu))$

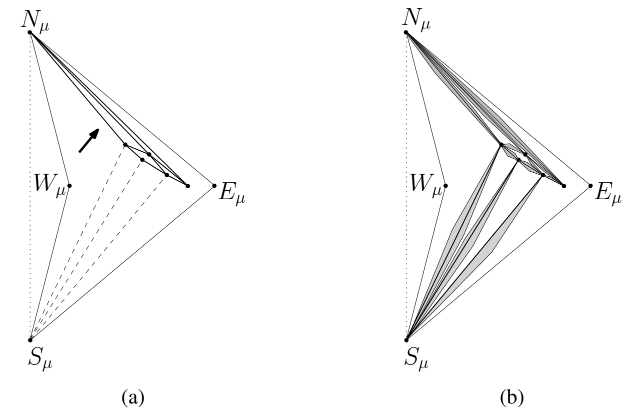


Figure 9.16: (a) A directional-scale applied to Γ^* in the direction perpendicular to $-d_N(\mu)$. (b) The drawing of $pert(\mu)$ obtained with the algorithm described in this section.

with either a boomerang or a diamond, depending on the type of node it represents, as follows.

Consider the pair of vertices x, y belonging to the subdivision of $skel(\mu)$ such that the range $range(P(x, y))$ of the monotone path $P(x, y)$ between them in $\Gamma'(skel(\mu))$ creates the largest angle $\angle(x, y)$ among all the pairs of vertices. Let $\gamma = \pi - \angle(x, y)$. Further, let δ be the smallest angle between two adjacent edges in $\Gamma(skel(\mu))$. Finally, let ϵ be the smallest angle between an edge incident to u and segment $\overline{N_\mu E_\mu}$.

For each node μ_i, with $i = 1, \dots, k$, represented by virtual edge (u_i, v_i), let N_{μ_i} and S_{μ_i} be the points where u_i and v_i have been drawn in $\Gamma(skel(\mu))$, respectively. Then, consider a boomerang $boom(\mu_i) = (N_{\mu_i}, E_{\mu_i}, S_{\mu_i}, W_{\mu_i})$ such that E_{μ_i} and W_{μ_i} determine $\beta_{\mu_i} + 2\alpha_{\mu_i} < \min\{\frac{\delta}{2}, \frac{\gamma}{2}, \frac{\epsilon}{2}\}$. If μ_i is a P-node with an edge between its poles, consider a diamond $diam(\mu_i)$ composed of two mirrored copies of $boom(\mu_i)$. Then, apply the inductive algorithm to μ_i, with poles u_i and v_i, and to either $boom(\mu_i)$ or $diam(\mu_i)$. See Figure 9.16(b).

Claim 9.4 *If μ is an R-node, the drawing Γ_μ constructed by algorithm* BICOFIXEDEMBEDDING *satisfies Properties (A)–(C).*

Proof: The fact that Γ_μ satisfies Property (A) depends on the monotonicity of $\Gamma'(skel(\mu))$ and on the fact that $\beta_{\mu_i} + 2\alpha_{\mu_i} < \frac{\gamma}{2}$, where $\gamma = \pi - \angle(x, y)$ and $\angle(x, y)$ is the largest angle created by the range $range(P(x, y))$ of the monotone path connecting any two vertices x and y. This implies that, for every monotone path between two vertices w_1 and w_2 in $\Gamma'(skel(\mu))$ (whose range creates an angle smaller than $\angle(x, y)$, by definition), it is possible to construct a monotone path in Γ_μ having range (strictly) smaller than $\pi - \angle(x, y) + \angle(w_1, w_2) \leq \pi$.

Property (B) follows from the planarity of the drawings of the child nodes, which is guaranteed by induction, and on the fact that the boomerangs and the diamonds containing such drawings do not overlap, due to the fact that $\beta_{\mu_i} + 2\alpha_{\mu_i} < \frac{\delta}{2}$. Also, as remarked before, all such boomerangs and diamonds are contained inside $boom(\mu)$, as no vertex has been placed on point E_μ.

Since for each edge $e \in \Gamma(skel(\mu))$ it holds $sl(-d_N(\mu)) - \frac{\alpha_\mu}{2} < sl(e) < sl(-d_N(\mu)) + \frac{\alpha_\mu}{2}$, if e is not incident to v, and $sl(d_S(\mu)) - \frac{\alpha_\mu}{2} < sl(e) < sl(d_S(\mu)) + \frac{\alpha_\mu}{2}$, if it is incident to e, and since $\beta_{\mu_i} + 2\alpha_{\mu_i} < \frac{\alpha_\mu}{2}$, Property (C) is also satisfied by Γ_μ. □

We state the main theorem of this section.

Theorem 9.3 *Algorithm* BICOFIXEDEMBEDDING *computes a straight-line embedding-preserving monotone drawing of every n-vertex biconnected embedded planar graph G in $O(n)$ time.*

Proof: The fact that algorithm BICOFIXEDEMBEDDING computes the required drawing follows from the fact that, as proved in the above discussion, Properties (A), (B), and (C) hold for each node of the SPQR-tree of G.

Concerning the linear bound on the computational complexity, first note that SPQR-trees can be constructed and handled in linear time [DT96a, DT96b, GM01]. Also, the computation of angles α_{μ_i} and β_{μ_i} at each step of the computation and, in the case of R-nodes, the construction of a convex drawing of $skel(\mu)$ and the operation directional-scale, can be performed in linear time in the size of the considered node, and hence in total linear time in the size of the graph. □

Chapter 10

Slanted Orthogonal Drawings

In this chapter[1] we focus on drawings of non-planar graphs. We introduce a new model in the context of non-planar orthogonal graph drawing that we call *slanted orthogonal graph drawing*. While in traditional orthogonal drawings each edge is made of alternating axis-aligned line-segments, in slanted orthogonal drawings intermediate diagonal segments on the edges are permitted, which allows for: (a) smoothening the bends of the produced drawing (as they are replaced by pairs of "half-bends"), and, (b) emphasizing the crossings of the drawing (as they always appear at the intersection of two diagonal segments). We present an approach to compute bend-optimal slanted orthogonal representations, an efficient heuristic to compute close-to-optimal slanted orthogonal drawings with respect to the total number of bends in quadratic area, and a corresponding LP formulation, when insisting on bend-optimality. On the negative side, we show that bend-optimal slanted orthogonal drawings may require exponential area.

The chapter is structured as follows. In Section 10.1 we give some preliminary definitions and introduce some related works. In Section 10.2 we present an approach to compute bend-optimal slog representations. Afterwards, we present a heuristic to compute close-to-optimal slog drawings, that require polynomial drawing area, based on a given slog representation. To compute the optimal drawing, we give a formulation as a linear program in Section 10.4. In Section 10.5 we show that the optimal drawing may require exponential area. In Sections 10.6 and 10.7, we present an experimental evaluation and some sample drawings of our algorithms, respectively.

[1]The contents of this chapter are joint work with Michael Bekos, Michael Kaufmann, Robert Krug, and Stefan Naher, appeared in [BKK+13] and have been submitted to journal.

10.1 Introduction

In this chapter, we introduce and study a new model in the context of non-planar orthogonal graph drawing: Given a graph G of max-degree 4, determine a drawing Γ of G in which (a) each vertex occupies a point on the integer grid and has four available *ports*, as in the ordinary orthogonal graph drawing model; (b) each edge is drawn as a sequence of alternating horizontal, vertical and diagonal segments; (c) a diagonal segment is never incident to a vertex (due to port constraints mentioned above); (d) crossings always involve diagonal segments; and (e) the minimum of the angles formed by two consecutive segments of an edge always is $135°$, which suggests that a bend in Γ is always incident to a diagonal segment and to either a horizontal or a vertical one. We refer to Γ as the *slanted orthogonal drawing* of G, or, shortly, *slog drawing*. For an example, refer to Figure 10.1(a). The corresponding slog drawing of this example is illustrated in Figure 10.1(b). This example indicates what we might expect from the new model: crossings on the diagonals are more visible than the corresponding ones in the traditional orthogonal graph drawing model and the use of area seems to be more effective.

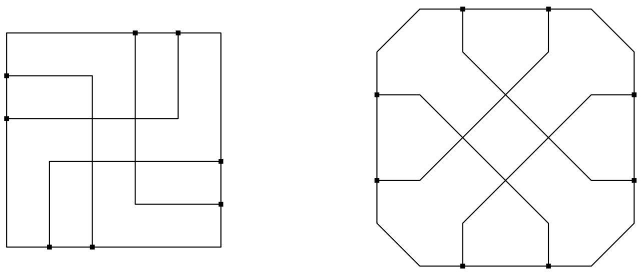


Figure 10.1: Traditional orthogonal (a) and slanted orthogonal (b) drawings of the same graph, assuming fixed ports.

Slog drawings generalize orthogonal drawings in the following sense: If the input graph G is planar, then any planar orthogonal drawing Γ of G can be transformed into a planar slog drawing Γ' of G, by replacing each bend of $90°$ of Γ by two "*half-bends*" of $135°$ in Γ', as illustrated in Figure 10.2[2]. The resulting drawings will be of improved readability and more aesthetic appeal, since bends, which negatively affect

[2]Potential crossings posed by the presence of half-bends can be avoided, if one scales Γ' by a factor of 2 and the diagonal segment defined by a pair of half-bends lies in a 1×1 box.

the quality of orthogonal drawings (as they interrupt the eye movement and require sharp changes of direction), are replaced by pairs of half-bends that have a smoother shape. In addition, slog drawings reveal the presence of crossings and help distinguishing them from vertices of the drawing, because crossings are defined by diagonal segments, while vertices are always incident to rectilinear segments.

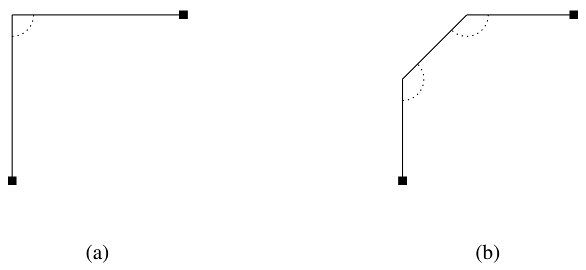


Figure 10.2: Replacing a 90° bend (a) by two half-bends of 135° (b).

10.1.1 Related Work

Orthogonal graph drawing has a long tradition, dating back to VLSI layouts and floorplanning applications [Lei80, TT89, Val81]. Formally, an *orthogonal drawing* of a graph of max-degree 4 is a drawing in which each edge is drawn as a sequence of alternating horizontal and vertical line-segments, and which is optimal under a prespecified optimization function which measures the niceness of the resulting drawing. Typical optimization functions include minimizing the used area [PT97, TT89], the total number of bends [FK95, GT01, Tam87] or the maximum number of bends per edge [BK94, LMS98]; for an overview see e.g. [EFK01].

For minimizing the total number of bends in orthogonal graph drawing Tamassia laid important foundations by the *topology-shape-metrics* (*TSM*) approach in [Tam87], that works in three phases. In the first *planarization* phase a "planar" embedding is computed for a given (non)planar graph by replacing edge crossings by dummy vertices (referred to as *crossing or c-vertices*). The output is called *planar representation*. In the next *orthogonalization* phase, angles and bends of the drawing are computed, producing an *orthogonal representation*. In the third *compaction* phase the coordinates for vertices and edges are computed. The core is a min-cost flow algorithm to minimize the number of bends in the second phase [CK12]. Note that the general problem of determining a planar embedding with the minimum number of bends is NP-hard [GT01].

Our model resembles an octilinear model as it is heavily used for example in the drawing of metro maps [NW11] but it is closer to the traditional orthogonal style. In particular, angles of $45°$ do not occur at all. Therefore, the complexity results for the octilinear model do not apply to our model.

Closely related to the problem we study is also the *smooth orthogonal drawing* problem [BKKS12], which asks for a planar drawing of an input planar graph of maximum degree 4, in which every edge is made of axis-aligned line-segments and circular-arcs with common horizontal or vertical tangents; the main goal is to determine such drawings with low edge complexity, measured by the number of line-segments and circular-arc segments forming each edge. Note that both approaches try to smoothen orthogonal drawings either by the usage of circular arc segments (smooth orthogonal drawings) or by replacing orthogonal bends by half-bends (slog drawings).

10.1.2 Preliminaries and Notation

For planar slog drawings, observe that the problem of minimizing the number of bends over all embeddings of an input planar graph of maximum degree 4 is NP-hard. This directly follows from [GT01], since the bends of a planar orthogonal drawing are in one to one correspondence with pairs of half-bends of the corresponding slanted orthogonal drawing. This negative result led us to adopt the TSM approach for our model. So, in the following, we assume that a planar representation of the input graph is given. Then, one can easily observe the following requirements: (a) all non-dummy vertices (referred to as *real* or *r-vertices*) use orthogonal ports and, (b) all c-vertices use diagonal ports. This ensures that the computed drawing will be a valid slog drawing that corresponds to the initial planar representation. Edges connecting real (crossing) vertices are referred to as *rr-edges* (*cc-edges*), and edges between r- and c-vertices as *rc-edges*.

We also use the notion of a *left* or *right* turn, which we define in the following.

Definition 10.1 *Let $e = (u, v)$ be an edge with at least one bend and let s and s' be two consecutive segments of e with b being the common bend of s and s'. Furthermore let ϕ be the angle formed by s and s' on their left side when moving along e from u to v. Edge e has a* left turn *on b if $\phi \leq 180°$, otherwise there is a* right turn *on b.*

10.2 Bend-Optimal Slanted Orthogonal Representations

In this section, we present an algorithm for computing a bend-optimal slog representation of an input plane graph of maximum degree 4. This algorithm is a modification of a well-known algorithm by Tamassia [Tam87] for computing bend-optimal orthogonal representations of plane graphs of maximum degree 4 by modeling the problem, as a min-cost flow problem on a flow network derived from the embedding of the graph. However, before we proceed with the detailed description of our modification, in Section 10.2.1 we briefly describe the algorithm of Tamassia. In Section 10.2.2, we describe in detail our modification. Section 10.2.3 presents properties of bend-minimal slog representations.

10.2.1 Preliminaries

A central notion to the algorithm of Tamassia [Tam87] is the *orthogonal representation*, which in a sense captures the "shape" of the resulting drawing, neglecting the exact geometry underneath. Typically, an orthogonal representation of a plane graph $G = (V, E)$ is an assignment of four labels to each edge $(u, v) \in E$; two for each direction. Label $\alpha(u, v) \cdot 90^\circ$ corresponds to the angle at vertex u formed by edge (u, v) and its next incident edge counterclockwise around u. Label $\beta(u, v)$ corresponds to the number of left turns of angle 90° along (u, v), when traversing it from u towards v. Clearly, $1 \leq \alpha(u, v) \leq 4$ and $\beta(u, v) \geq 0$. Since the sum of angles around a vertex equals to 360°, it follows that for each vertex $u \in V$, $\sum_{(u,v)\in N(u)} \alpha(u, v) = 4$, where $N(u)$ denotes the neighbors of u. Similarly, since the sum of the angles formed at the vertices and at the bends of a bounded face f equals to $180^\circ(p(f) - 2)$, where $p(f)$ denotes the total number of such angles, it follows that $\sum_{(u,v)\in E(f)} \alpha(u, v) + \beta(v, u) - \beta(u, v) = 2a(f) - 4$, where $a(f)$ denotes the total number of vertex angles in f, and, $E(f)$ the directed arcs of f in its counterclockwise traversal. If f is unbounded, the respective sum is increases by eight. It is known that two orthogonal drawings with the same number of bends at each edge have the same orthogonal representation.

There is a nice correspondence between the min-cost network flow formulation of Tamassia and the underlying orthogonal representation with minimum number of bends of the input plane graph. In the flow network, one can think that each unit of flow corresponds to a 90° angle. Then, the vertices (*vertex-nodes*; *sources*) supply four units of flow each, which have to be consumed by the faces (*face-nodes*; *sinks*). Each face f demands $2a(f) - 4$ units of flow (increased by eight if f is unbounded). The relation now seems clear. To maintain the properties described above each edge from a vertex-node to a face-node in the flow network is equipped with a capacity of 4 and a minimum flow of 1, while an edge between adjacent faces has infinite capacity, no lower bound but each unit of flow through it introduces a respective unit cost. The total cost is actually the total number of bends along the respective edge. Hence, the min cost flow solution corresponds to a representation of the plane graph with minimum total number of bends.

10.2.2 Modifying the Flow Network

We are now ready to present how to modify the algorithm of Tamassia, in order to obtain a slog representation of an input plane graph G with minimum number of half-bends. Recall that G contains two types of vertices, namely real and crossing vertices. Real (crossing, respectively) vertices use orthogonal (diagonal, respectively) ports. Observe that a pair of half-bends on an rr-edge of a slog drawing corresponds to a bend of an orthogonal drawing. The same holds for half-bends on cc-edges. However, an rc-edge must switch from an orthogonal port (incident to the r-vertex) to a diagonal port (incident to the c-vertex). This implies that each rc-edge has at least one half-bend.

Consider an rc-edge (v_r, v_c) incident to faces f and g (see Figure 10.3) and assume that the port of real vertex v_r is fixed. Depending on the (diagonal) port on the crossing vertex v_c we obtain two different representations with the same total number of bends. To model this "free-of-cost" choice, we introduce an edge into the flow network connecting f and g with unit capacity and zero cost, i.e., through this edge just one unit of flow can be transmitted and this is for free. Hence, the first half-bend of each rc-edge is free of cost, as desired. For consistency we assume that, if in the solution of the min cost flow problem there is no flow over (f, g), then there exists a left turn from the real to the crossing vertex on the bend before the crossing; otherwise a right turn, as illustrated in Figure 10.3.

10.2.3 Existence and Properties of Bend-Optimal Slanted Orthogonal Representations

In the following we present properties of optimal slog representations. We prove that, for a planarized graph G, the computation of a slog representation with minimum number of half-bends that respects the embedding of G is always feasible. Then, we present an upper bound for the number of half-bends in optimal slog representations. In the following we assume that, together with a planarization, the embedding is also given.

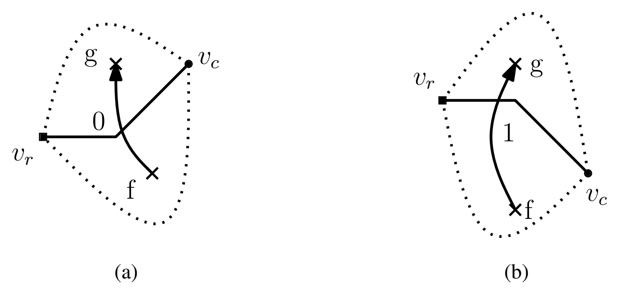


Figure 10.3: Two configurations corresponding to zero (a) or one (b) unit of flow over an rc-edge; f and g are the two adjacent faces.

Theorem 10.1 *Let G be a plane (or planarized) graph with maximum degree 4. A slog representation of G with the minimum number of half-bends can be computed efficiently.*

Proof: The idea is to use a reduction to Tamassia's network flow algorithm. In particular, since the original flow network algorithm computes a (bend-optimal) orthogonal representation for the input plane graph, our algorithm will also compute a slog representation. In the following, we prove that this representation is also bend-optimal.

Assume that we are given an orthogonal representation F. We can uniquely convert F into a slog representation $S(F)$ by turning all crossing vertices counterclockwise by $45°$. More precisely, the last segment of every rc-edge before the crossing vertex will become a left half-bend. Furthermore, every orthogonal bend is converted into two half-bends, bending in the same direction as the orthogonal bend (see Figure 10.2). Note that the left half-bends at the crossings might neutralize with one of the half-bends originating from an orthogonal bend, if the orthogonal bend is turning to the right (see Figure 10.4). In this case, only the second one of the right half-bends remains. Note that this is the only possible saving operation. Therefore, since the number of rc-edges is fixed from the given embedding, a slog representation with minimum number of half-bends should minimize the difference between the number of orthogonal bends of F and the number of first right-bends on rc-edges. However, this is exactly what is done by our min cost flow network formulation, as the objective is the minimization of the total number of bends in F without the first right-bends on rc-edges. □

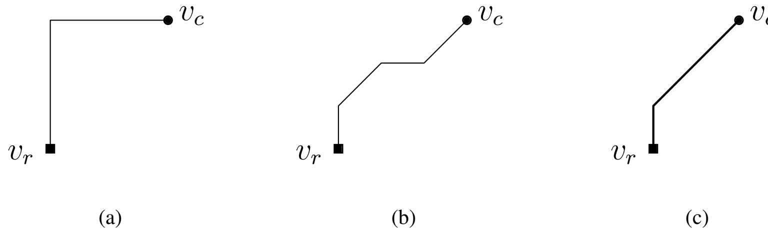


Figure 10.4: Illustration for Theorem 10.1. An orthogonal representation (a) of a bent rc-edge can be converted into a slog representation (b). (c) Two superfluous half-bends can be eliminated.

This constructive approach can also be reversed such that for each slog representation S, we can construct a unique orthogonal representation $F(S)$. Clearly, $F(S(F)) = F$ and $S(F(S)) = S$. Note that this is true only for bend-minimal representations. If this is not the case, then one has to deal with staircases of subsequent bends; a case that cannot occur in min-cost flow computations. From the construction, we can also derive the following.

Corollary 10.1 *Let $S(F)$ be a slog representation and F a corresponding orthogonal representation. Let b_S, rb_S and rc_S be the number of half-bends, the number of first right-bends on rc-edges and the number of rc-edges in $S(F)$. Let also b_F be the number of orthogonal bends in F. Then, $b_S = 2 \cdot (b_F - rb_S) + rc_S$.*

The following theorem gives an upper bound for the number of half-bends in optimal slog representations.

Theorem 10.2 *The number of half-bends of a bend-minimal slog representation is at least twice the number of bends of its corresponding bend-minimal orthogonal representation.*

Proof: Bends of a bend-minimal orthogonal representation correspond to pairs of half-bends on cc- and rr-edges of a bend-minimal slog representation. So, in this case, the claim holds with equality. But for rc-edges we need a different argument.

Let $\mathcal{C}$ be a maximal component spanned by cc-edges. By definition, all edges that have exactly one endpoint in $\mathcal{C}$ are rc-edges. Now, observe that rc-edges can be split into several cycles around components of crossings. However, these cycles are independent. Now, consider such a cycle C of length k. Clearly, there should be k

first half-bends on the rc-edges in the slanted representation, since each rc-edge has to bend at least once.

In the corresponding orthogonal representation, the second and third bend of each rc-edge correspond to pairs of half-bends on the same edge in the slog representation. Similarly, in the orthogonal representation the first orthogonal left-bend of each rc-edge corresponds to the second and third left half-bend of the same edge in the slog representation. So, the only bends that have not been paired (and subsequently have no correspondence) are the first right-bends on rc-edges.

We now claim that in any bend-minimal orthogonal representation, there exist at most $\frac{k}{2}$ first right-bends on the edges of cycle C. For a proof by contradiction, assume that in a bend-minimal orthogonal representation, there exist $r > \frac{k}{2}$ first right-bends on the edges of C. If we send the flow along C in reverse direction, we decrease the number of right-bends by r and increase the number of left-bends by $k-r$. Hence, the total number of bends decreases, which shows that the input orthogonal representation was not minimal; a contradiction.

From the claim, it follows that the number of first right-bends in the orthogonal representation is at most half of the number of first half-bends of cycle C (in the corresponding slog representation), which concludes the proof since all other half-bends come in pairs and have their correspondences. □

10.3 A Heuristic to Compute Close-to-Optimal Slanted Orthogonal Drawings

In this section, we present a heuristic which, given an optimal slog representation, computes an actual drawing, which is close-to-optimal with respect to the total number of bends and requires quadratic area. This is a quite reasonable approach, since insisting on optimal slog drawings may result in exponential area requirements, as we will shortly see in Section 10.5. The basic steps of our approach are outlined in Algorithm `Spoon Based`. In the following, we describe them in detail.

In Step 1 of Algorithm `Spoon Based`, we compute an orthogonal drawing Γ based on the input slog representation. If there is flow on an edge e connecting faces f_i and f_j that we added, we treat it as if it was flow on the other edge connecting f_i and f_j that was part of the flow network of the original algorithm. With this we get a flow that is still valid and corresponds to an orthogonal representation for which the algorithm of Tamassia [Tam87] can compute a drawing. In the next step, we replace all orthogonal bends with pairs of half-bends. In Step 3 of Algorithm `Spoon Based`, we connect r-vertices with c-vertices by replacing the segment incident to the c-vertex of each rc-edge by a gadget, which we call *spoon* due to its shape (see Figure

Algorithm `Spoon Based`

Require: A slog representation S of a given plane graph G.
Ensure: A slog drawing Γ_s of G.

```
1. Compute an orthogonal drawing Γ based on S
2. Replace each orthogonal bend by 2 half-bends          /* see Figure 10.2 */
3. Fix ports on rc-edges using the spoon gadget      /* see Figure 10.5(a) */
4. Apply cuts to fix ports on cc-edges          /* see Figures 10.5(b)-(c) */
5. Optimize the number of rc half-bends         /* see Figures 10.6(a)-(b) */
6. Optimize the number of cc half-bends                  /* see Figure 10.7 */
7. Heuristically compact the drawing
```

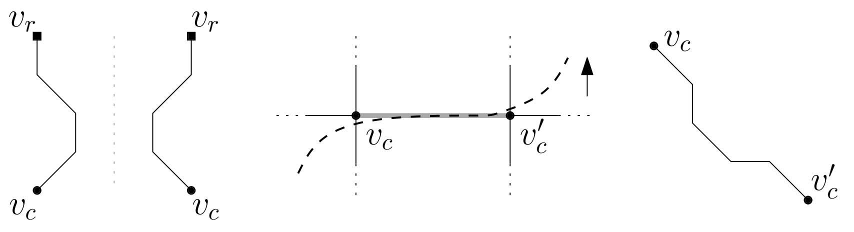
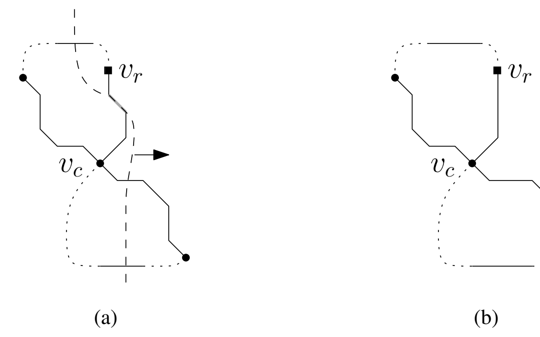


Figure 10.5: (a) Illustration of the spoon gadget. (b) The orthogonal input can be transformed into a slog by translating upwards everything above the dashed cut. (c) The result contains 4 half-bends.

10.5(a)). This gadget allows us to switch between orthogonal and diagonal ports on an edge. Note that the input slog representation specifies the ports on all vertices, thereby defining which configuration is used.

In order to fix the ports of cc-edges (which still use orthogonal ports), we employ appropriate cuts[3] (Step 4 of Algorithm `Spoon Based`). A *cut*, for us, is either (i) an x-monotone continuous curve that crosses only vertical segments and divides the current drawing into a top and a bottom part (*horizontal cut*), or, (ii) a y-monotone continuous curve that crosses only horizontal segments and divides the current drawing into a left and a right part (*vertical cut*). Observe that in order to apply a horizontal (vertical, respectively) cut, we have to ensure that each edge crossed by the cut has

[3]A *cut* is a standard tool to perform stretchings in orthogonal drawings, see e.g. [FHK98].



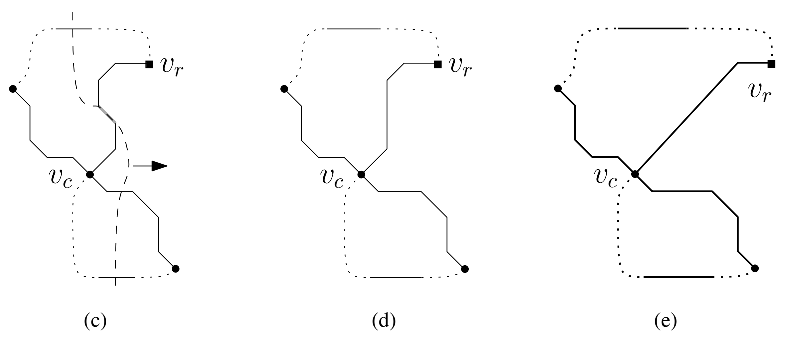


Figure 10.6: Saving bends on rc-edges: A cut through an rc-edge (v_r, v_c) leads to two half-bends reduction; the optimal may require four half-bends reduction.

at least one vertical (horizontal, respectively) segment. This holds before the introduction of the spoons, as Γ is an orthogonal drawing. We claim that this also holds when all spoons are present. This is because a spoon replacing a horizontal (vertical, respectively) segment has two horizontal (vertical, respectively) segments. To fix a horizontal cc-edge (v_c, v_c') with v_c being to the left of v_c' in the drawing, we first momentarily remove this edge from the drawing. Then we use a horizontal cut which from left to right passes exclusively through vertical segments. It starts in the outer face and continues either up to the face below (v_c, v_c'), then to the face above and from there again to the outer face. Or it continues up to the face above (v_c, v_c'), then to the face below and from there to the outer face (see Figure 10.5(b)).

Our choice depends on the input slog representation that specifies the ports on each crossing vertex. The result of such a cut is depicted in Figure 10.5(c) and has a new horizontal and a new vertical segment that replaces edge (v_c, v'_c). The first (second, respectively) one is necessary for potential future vertical (horizontal, respectively) cuts. Similarly, we cope with cc-edges with bends by applying the same technique only to the first and last segments of the edge.

The resulting slog drawing has three additional half-bends for each rc-edge (the spoon gadget adds three half-bends; one is required) and four additional half-bends for each cc-edge (none is required), with respect to the ones suggested by the input representation. With similar cuts as the ones described above, we can save two half-bends for each rc-edge, by eliminating the diagonal segment of the spoon gadget (Step 5 of Algorithm `Spoon Based`). Our approach is illustrated in Figures 10.6(a) and 10.6(b). Observe that in this case the cut simply requires the removal of the diagonal segment that is to be eliminated and not the whole edge. The result is optimal for bend-less rc-edges (see Figure 10.6(b)). However, for rc-edges with bends (see Figure 10.6(c)), our approach guarantees two half-bends reduction (see Figure 10.6(d)), while in the optimal case four half-bends could be removed (see Figure 10.6(e)). Observe that the rectilinear segments of the edge are not affected, in order to be able to apply future cuts.

As already stated, each cc-edge admits four additional half-bends (none is required). It is always possible to remove two of them (Step 6 of Algorithm `Spoon Based`) by applying a local modification as depicted in Figure 10.7. If for example the horizontal part of such an edge is longer than the vertical one, a shortcut like the one in the left part of Figure 10.7(a) can be applied. Note that this operation does not require any cuts. If the horizontal and the vertical segments of the cc-edge have the same length, then all four half-bends can be saved; see Figure 10.7(c).

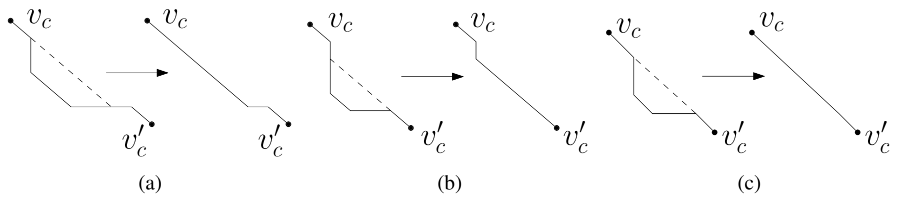


Figure 10.7: Saving bends on cc-edges by a local operation.

Once the operations we described above are applied, the drawing will contain zero additional half-bends on rr-edges and bend-less rc-edges and at most two additional half-bends on each cc-edge and each rc-edge with bends, with respect to the input representation. Note that in order to apply our technique we need to scale up the initial drawing by a factor of 5 at the beginning of our algorithm, to provide enough space for additional half-bends. In subsequent steps, the cuts increase the drawing area. However, since each cut implies a constant factor increment to the drawing area and each edge yields at most one cut, the total drawing area asymptotically remains quadratic. Heuristically, we can further reduce it by contracting along horizontal and vertical cuts as long as no crossings occur (Step 7 of Algorithm `Spoon Based`); however, this post-processing does not result into asymptotically smaller area. The following theorem summarizes our approach.

Theorem 10.3 *Given a slog representation of a planarized graph G of maximum degree 4, we can efficiently compute a slog drawing requiring $O(n^2)$ area with (i) optimal number of half-bends on rr- and bend-less rc-edges and (ii) at most two additional half-bends on cc edges and rc-edges with bends.*

10.4 A Linear Program for Computing Optimal Drawings

In this section, we develop a Linear Program (LP) which, given an optimal slog representation S of a plane graph G, computes an actual drawing Γ, which is optimal with respect to the total number of bends; if one exists. Before we proceed with the description of our linear program, we mention that despite the fact that every experiment we made on random and crafted graphs led to a feasible solution, we could not prove the feasibility of the linear program.

10.4.1 The Core of the Linear Program

Initially, we appropriately augment graph G and obtain a new graph that is a subdivision of G and has at most one half-bend on each edge. More precisely, let (u, v) be an edge of G with more than two half-bends (as defined by the slog representation S). Let $\langle b_1, b_2, \ldots, b_k\rangle$, $k \geq 2$, be the half-bends of edge (u, v) and assume without loss of generality that $b_1, b_2, \ldots, b_k$ appear in this order along the edge (u, v), when traversing (u, v) from vertex u towards vertex v. We first consider the case where vertex u is a real vertex. In this case, we add a new crossing vertex w in G and then we replace the edge (u, v) of G with the edges (u, w) and (w, v). The first half-bend b_1 of the edge (u, v) is assigned to the edge (u, w), while the remaining half-bends $\langle b_2, \ldots, b_k\rangle$ of the edge (u, v) are assigned to the edge (w, v). The case where vertex u is a crossing vertex is treated analogously, with the only exception that in this particular case vertex w would have been a real vertex. Then, it is clear that if we apply the procedure that we just described on each edge of G with more than two half-bends (as long as there exist such edges), then we will obtain an augmented graph, say G_{aug}, that is clearly a subdivision of G and has at most one half-bend on each edge, as desired. Furthermore, neither the type of each new vertex nor its ports are arbitrarily chosen, as they depend on the slopes of its incident segments. This implies a new slog representation, say S_{aug}, for G_{aug}.

Now observe that each face f of G has a corresponding face f' in G_{aug} such that: (i) the vertices of G_{aug} incident to face f' are the same as the ones incident to face f of G, plus the ones from the subdivision; and (ii) the sequence of slopes assigned to the segments bounding f' is the same as the ones of the segments bounding f in G. Hence, a drawing Γ_{aug} of G_{aug} realizing the slog representation S_{aug} is also a drawing Γ of G realizing the slog representation S, where subdivided edges are routed as their corresponding paths in G_{aug}.

We are now ready to describe our linear program, which computes a drawing Γ_{aug} of G_{aug} realizing the slog representation S_{aug}. For each vertex u of G_{aug}, we introduce a pair of variables x_u and y_u that corresponds to the coordinates of vertex u on the plane. Then, for each edge (u, v) of G_{aug}, we define a pair of constraints, depending on the type of vertices u and v (i.e., real or crossing vertices). The detailed list of constraints is given in Figure 10.8.

In order to obtain "compact" drawings, we indirectly minimize the area by minimizing the total edge length. In particular, this is our objective function. Note that the slopes of the segments allow us to express the Euclidean length of each edge as a linear function. As an example, the length of the edge depicted in the first cell of Figure 10.8(c) is defined as $(\sqrt{2} - 1) \cdot (y_u - y_v) + x_v - x_u$.

10.4.2 Addressing Planarity Issues

The linear program, as described so far, models the shape of the edges (and subsequently the shape of the faces) and the relative positions between pairs of adjacent vertices. Since there are no constraints among non-adjacent vertices, it is highly possible that the resulting drawing is non-planar. We provide an example in Figure 10.9(a), where the relative positions between vertices (i) v_r and v_c, and, (ii) v_r and v'_c are not defined by the liner program, yielding to a (potential) crossing situation. To cope with this problem, unfortunately, we cannot follow an approach similar to the one that Tamassia suggests in his original algorithm (i.e., he "splits" all non-rectangular faces into rectangular ones), since in our case a face is not necessarily rectilinear.

In order to describe our approach to ensure that each face is drawn planar, we first introduce some necessary terminology. We distinguish two types of corners of a face in a slog representation; *vertex-corners* (or simply vertices) and *bend-corners* (or simply bends). With respect to a face, a corner is either *convex*, if the inner angle

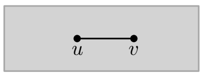
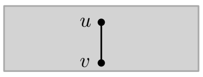
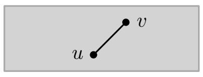
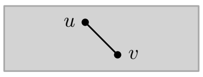
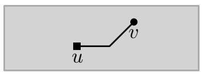
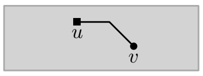
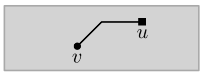
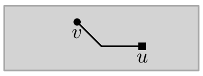
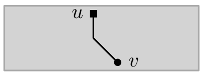
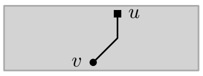
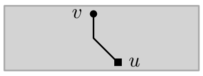
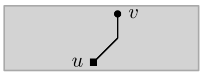
(a) rr-edges		(b) cc-edges	
 $y_u = y_v$ $x_v - x_u \geq 1$	 $x_u = x_v$ $y_v - y_u \geq 1$	 $y_u - y_v = x_v - x_u$ $y_u - y_v \geq 1$	 $y_v - y_u = x_v - x_u$ $y_v - y_u \geq 1$
(c) rc-edges			
 $x_v - x_u \geq y_u - y_v + 1$ $y_u \geq y_v + 1$	 $x_v - x_u \geq y_v - y_u + 1$ $y_v \geq y_u + 1$	 $x_u - x_v \geq y_v - y_u + 1$ $y_v \geq y_u + 1$	 $x_u - x_v \geq y_u - y_v + 1$ $y_u \geq y_v + 1$
 $y_v - y_u \geq x_v - x_u + 1$ $x_v \geq x_u + 1$	 $y_v - y_u \geq x_u - x_v + 1$ $y_v \geq y_u + 1$	 $y_u - y_v \geq x_u - x_v + 1$ $x_u \geq x_v + 1$	 $y_u - y_v \geq x_v - x_u + 1$ $x_v \geq x_u + 1$

Figure 10.8: The list of constraints used by the linear program for (a) rr-edges, (b) cc-edges and (c) rc-edges, assuming that the y-axis points downwards.

is $\leq 135°$, or *non-convex* otherwise[4]. Hence, there are four possible types of corners in total: convex vertex-corner, convex bend-corner, non-convex vertex-corner, non-convex bend-corner. The *configuration* of a corner describes the shape of the corner by the pair of orientations of its two incident segments in the order they are visited by a counterclockwise traversal of the corresponding face. Possible *orientations* are horizontal (h), vertical (v), diagonal-up (du), and diagonal-down (dd). For example, the configuration of the bend-corner incident to segments s' and s'' of Figure 10.10(a) is given by du-h. The *type* of a configuration describes the corresponding corner in a more general way by just distinguishing between orthogonal (o) or diagonal (d) orientations. In the example of Figure 10.10(a), the configuration of the bend-corner incident to segments s' and s'' is of type *d-o*. We next define the notions of a *split-edge* and an *almost-convex face*, that are both central in our approach.

Definition 10.2 *For a given face f, a* split-edge *is an edge that:*

[4]We ignore vertices and bends on corners that form 180° angles, since by construction they are always aligned with their neighbors.

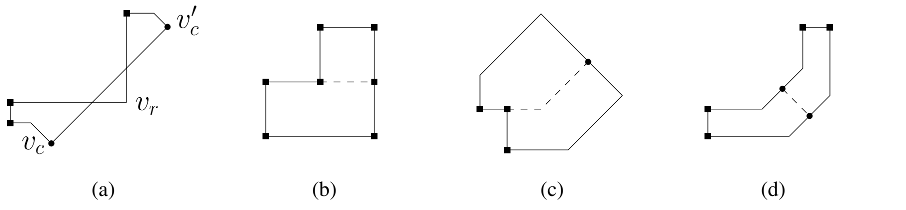


Figure 10.9: (a) A non-planar face. (b) Split-edge (vertex). (c) Split-edge (vertex) with half-bend. (d) Split-edge (bend). In all figures, real (crossing, respectively) vertices are drawn as squares (disks, respectively); split-edges are drawn dashed.

- *is bend-less and connects a non-convex vertex-corner v with a new vertex that we introduce by subdividing a side parallel to one of the edges incident to v (see Figure 10.9(b)).*
- *or, has a half-bend and connects a non-convex vertex-corner v with a new vertex that we introduce by subdividing a diagonal side of f (see Figure 10.9(c)).*
- *or, is a bend-less edge that connects two new vertices that we introduce by subdividing two parallel edges, when one of them is incident to a non-convex bend-corner (see Figure 10.9(d)).*

Definition 10.3 *A face is* almost-convex *if it does not contain any non-convex vertex-corners and no split-edge exists that separates the face into two non-convex faces.*

First, we make all faces almost-convex (by further augmenting our graph). Later, we will show that the linear program will always compute a planar drawing if all faces are almost-convex.

A non-convex vertex-corner is *eliminated* by introducing a new split-edge (corresponding to new constraints in the linear program) as shown in Figure 10.9(b). When there is no parallel side to one of the segments incident to the vertex-corner we introduce a split-edge with a half-bend, as illustrated in Figure 10.9(c). It is important to note that the elimination of a non-convex vertex-corner does not introduce new ones. Hence, all of them can be eliminated sequentially by appropriately adopting one of the two approaches described above.

In order to eliminate a non-convex bend-corner of a face that is not almost-convex, we search for a split-edge (again corresponding to new constraints in the linear program) that yields two non-convex faces. Such a split-edge is illustrated in Figure 10.9(d).

We will appropriately introduce such split-edges until all faces are almost-convex (without introducing non-convex vertex-corners). To prove that it is always feasible to make all faces almost-convex, we give the following lemma.

Lemma 10.1 *Let s' and s'' be two segments of a face f incident to a non-convex bend-corner. Face f contains a segment $s \notin \{s', s''\}$ that is parallel to either s' or s''.*

Proof: For a proof by contradiction, we assume that there is no segment of face f parallel to s' and s''. Without loss of generality, we further assume that s' is a horizontal segment and s'' is a diagonal segment of positive slope; see Figure 10.10(a). The cases, where s' is a vertical segment and/or s'' is a diagonal segment of negative slope, are analogous (see Figures 10.10(b) to 10.10(d)). Let $p_{s'}$ and $p_{s''}$ be the end-points of segments s' and s'', respectively, which are not identified with the non-convex bend-corner incident to both s' and s''. Since f is a face, there exists a polygonal chain of segments of f connecting $p_{s'}$ and $p_{s''}$. In our drawing model, such a chain consists of horizontal, vertical and diagonal segments. Now observe that a horizontal or a positively-sloped diagonal segment of the chain connecting $p_{s'}$ and $p_{s''}$ is parallel to s' or s'', respectively, which contradicts our initial assumption that there is no segment of face f parallel to s' and s''. Hence, the polygonal chain connecting $p_{s'}$ and $p_{s''}$ consists of vertical and negatively-sloped diagonal segments, which is a contradiction since $p_{s'}$ and $p_{s''}$ cannot be connected by such a chain, without forming an angle of $45°$ at a corner of f (a situation that is not allowed by our drawing model). □

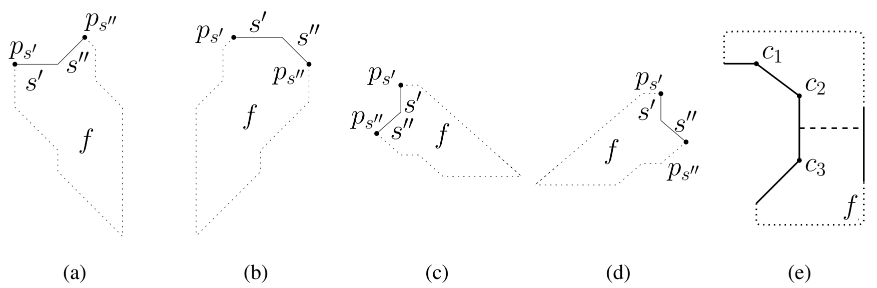


Figure 10.10: (a)-(d) Different configurations used in the proof of Lemma 10.1. (e) Configuration used in the proof of Lemma 10.2.

From Lemma 10.1, it follows that, for a non-convex bend-corner of a face f, there is a split-edge emanating from one of its incident segments towards to a parallel segment of face f. If f is not almost-convex (and contains no convex vertex-corners) and this edge is carefully selected such that it yields exactly two non-convex "subfaces", say f' and f'', of face f, then it is not difficult to see that both f' and f'' have fewer non-convex bend-corners than f. In addition, no convex vertex-corners are introduced. This implies that if one recursively applies this procedure to f' and/or f'' (if either of these is not almost-convex), f will eventually be split into a particular number of "subfaces" that are all almost-convex. In addition, it is not difficult to see that all additional edges, that are required to make all faces almost-convex can be expressed by using the original set of constraints of our linear program. So, it now remains to prove that almost-convex faces are drawn planar. To do so, we give the following lemmas.

Lemma 10.2 *An almost-convex face f has at most two consecutive non-convex bend-corners.*

Proof: Assume to the contrary that f has three consecutive non-convex bend-corners, say c_1, c_2 and c_3; see Figure 10.10(e). Assume that c_1, c_2 and c_3 appear in this order in the counterclockwise traversal of face f. By Lemma 10.1, there exists a segment of f that is parallel to one of the segments incident to c_2. This implies that, there exists a split-edge that partitions f into two non-convex faces; one containing c_1 and one containing c_3, which is a contradiction since f is almost-convex. □

Lemma 10.3 *An almost-convex face has at most two non-convex bend-corners.*

Proof: In the proof, we use the notion of a configuration. More precisely, we assume to the contrary that an almost-convex face f contains at least three non-convex bend-corners c_1, c_2 and c_3 and distinguish four cases. In our case analysis, we denote by $s^1_{c_i}$ and $s^2_{c_i}$ the segments incident to corner c_i and assume the $s^1_{c_i}$ precedes $s^2_{c_i}$ in the clockwise traversal of face f, $i = 1, 2, 3$.

Case 1: *Two of these non-convex bend-corners have the same configuration*; see Figures 10.11(a) and 10.11(b) for an illustration. By Lemma 10.1, there exists a parallel segment to either $s^1_{c_1}$ or $s^2_{c_1}$, and thereby to either $s^1_{c_2}$ or $s^2_{c_2}$. In both cases, one of the split-edges separates c_1 from c_2, so that the resulting faces are both non-convex. Hence, f is not almost-convex; a contradiction. So, in the following cases we assume that c_1, c_2 and c_3 are of different configurations.

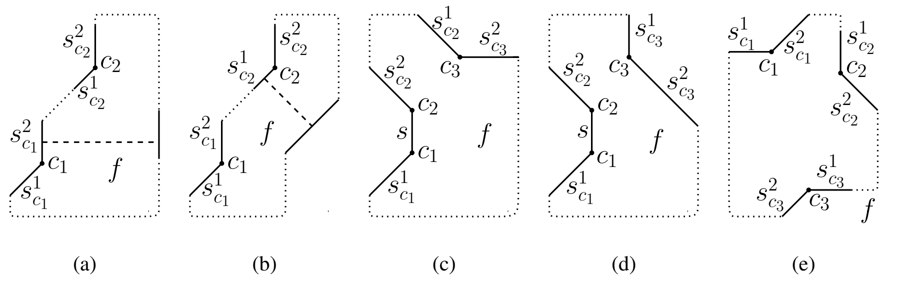


Figure 10.11: Different configurations used in the proof of Lemma 10.3.

Case 2: *Corners c_1 and c_2 are consecutive corners of f and the first segment of c_3 is parallel to the second segment of c_2*; see Figure 10.11(c) for an illustration. We denote by s the segment that is incident to both c_1 and c_2 (i.e., $s = s^2_{c_1} = s^1_{c_2}$) and first assume that c_1 and c_3 are of the same configuration. In order to close the face there has to be a segment that is parallel to either s or $s^2_{c_2}$ that is not $s^1_{c_3}$, thereby allowing a split-edge that separates either c_1 from c_2 and c_3, or, c_1 and c_2 from c_3. The resulting faces are both non-convex. Hence, f is not almost-convex; a contradiction. The case where c_3 has the opposite configuration of c_1 is analogous.

Case 3: *Corners c_1 and c_2 are consecutive and the second segment of c_3 is parallel to the second segment of c_2*; see Figure 10.11(d) for an illustration. Again, we denote by s the segment that is incident to both c_1 and c_2 (i.e., $s = s^2_{c_1} = s^1_{c_2}$) and assume that c_1 and c_3 are of opposite types of configuration. In this case there is a split-edge between segments $s^2_{c_2}$ and $s^2_{c_3}$ thereby separating c_1 and c_2 from c_3 and resulting in two non-convex faces. Hence, f is not almost-convex; a contradiction. The case that c_3 has the same type of configuration to c_1 is analogous.

Case 4: *Corners c_1, c_2 and c_3 are pairwise non-consecutive*; see Figure 10.11(e) for an illustration. Since there are only two types of diagonals, at least two non-convex corners, say c_1 and c_3, are of the same type. Since they are forced to have opposite configurations (*d-o* or *o-d*) a split-edge between those two parallel diagonals would separate the two respective corners, resulting in two non-convex faces. Hence, f is not almost-convex; a contradiction.

The proof is completed by the observation that one of these four cases will always apply to every almost-convex face with more than two non-convex bend-corners. □

Lemma 10.4 *An almost-convex face is always drawn planar.*

Proof: A face that is convex is drawn planar by definition. Let f be an almost-convex face. By Lemma 10.3, f has at most two non-convex bend corners. In the case where f has exactly one non-convex bend-corner, f is drawn planar, since it cannot have an even number of crossings without violating the port constraints. Consider now the more interesting case where face f has exactly two non-convex bend-corners, say c_1 and c_2. We denote by $s^1_{c_i}$ and $s^2_{c_i}$ the segments incident to corner c_i and assume the $s^1_{c_i}$ precedes $s^2_{c_i}$ in the clockwise traversal of face f, $i = 1, 2$. We distinguish the following cases:

Case 1: *Corners c_1 and c_2 are consecutive*; see Figure 10.12(a) for an illustration. In this case, there is a segment, say s, that is incident to both c_1 and c_2 (i.e., $s = s^2_{c_1} = s^1_{c_2}$). If there is a segment of f parallel to s, then there exists a split-edge separating f into two non-convex subfaces; one containing c_1 and one containing c_2 (see Figure 10.12(a)). Hence, f is not almost-convex. It follows that there is no segment of f that is parallel to s. By Lemma 10.1, there exist parallel segments to the other two segments that are incident to c_1 and c_2 (see Figure 10.12(b)). However, since f is almost-convex, a "split-edge" connecting the respective parallel segments would result in at least one convex face. We can move these "split-edges" arbitrary close to c_1 and c_2, so that they separate f into three convex regions. Since convex regions are drawn convex and hence planar by definition, no crossing can occur.

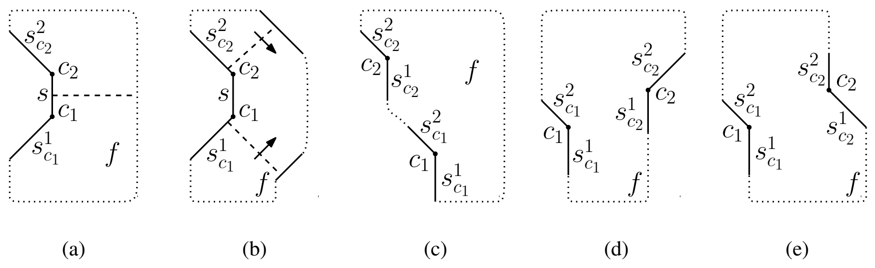


Figure 10.12: Different configurations used in the proof of Lemma 10.4.

Case 2: *Corners c_1 and c_2 have the same configuration and orientation*; see Figure 10.12(c) for an illustration. This particular case is identical to Case 1 of Lemma 10.3 and therefore cannot occur.

Case 3: *Corners c_1 and c_2 have opposite configuration (meaning that they are made of the same orthogonal and diagonal part but in different orders) and orientation*; see Figure 10.12(d) for an illustration. Since the number of crossings that occur has to be even (otherwise ports would be violated), and the only way to have two crossings requires that one of the convex regions is drawn non-convex, this situation cannot introduce any crossings.

Case 4: *Corners c_1 and c_2 have the same configuration but opposite orientations*; see Figure 10.12(e) for an illustration. In this case, it is not difficult to see that there exists a split-edge between the two orthogonal or the two diagonal segments incident to c_1 and c_2, separating them into two non-convex subfaces, so f cannot be almost-convex.

The proof is completed by the observation that one of these four cases will always apply to an almost-convex face with exactly two non-convex bend-corners. □

10.5 Area Bounds

Slog drawings have aesthetic appeal and seem to improve the readability of non-planar graphs, when compared to traditional orthogonal drawings. However, in this section we show that such drawings may require increased drawing area. Note that most of the known orthogonal drawing algorithms require $O(n) \times O(n)$ area. The situation is different if one insists on slog drawings of optimal number of bends. As the following theorem asserts, the area penalty can be exponential.

Theorem 10.4 *There exists a graph G whose slanted orthogonal drawing Γ of minimum number of bends requires exponential area, assuming that a planarized version σ of the resulting drawing is given.*

Proof: The planarized version σ of G is given in Figure 10.13(a) and consists of $n+1$ layers $L_0, L_1, \ldots, L_n$. Layer L_0 is the square grid graph on 9 vertices. Each layer L_i, $i = 1, 2, \ldots, n$, is a cycle on 20 vertices. Consecutive layers L_{i-1} and L_i, $i = 1, 2, \ldots, n$, are connected by 12 edges which define 12 crossings. Hence, G consists of $20n + 9$ vertices and $32n + 13$ edges that define $12n$ crossings.

A slog drawing Γ of G with minimum number of bends derived from σ ideally introduces (i) no bends on crossing-free edges of σ, and, (ii) two half-bends in total

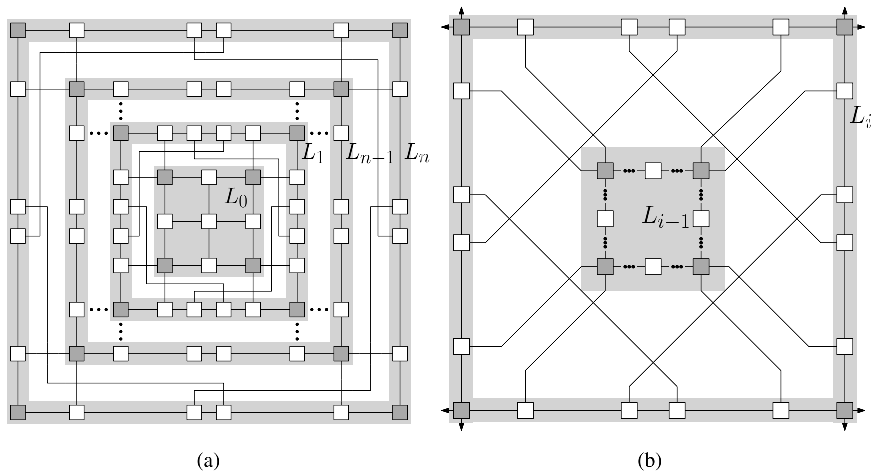


Figure 10.13: (a) A planarized version σ of a graph G. (b) Edges involved in crossings in σ contribute two half-bends.

for each rc-edge. Now observe that at each layer there exist four vertices, that have two ports pointing to the next layer (gray-colored in Figure 10.13(a)). This together with requirements (a) and (b) suggests that the vertices of each layer L_i should reside along the edges of a rectangle, say R_i, such that the vertices of L_i whose ports point to the next layer coincide with the corners of R_i, $i = 0, 1, 2, \dots, n$ (with the only exception of the “innermost” vertex of L_0; in Figure 10.13(b), R_i is identified with cycle L_i). Hence, the routing of the edges that connect consecutive layers should be done as illustrated in Figure 10.13(b). Since L_0 is always drawable in a 3×3 box meeting all requirements mentioned above, and, σ is highly symmetric, we can assume that each R_i is a square of side length w_i, $i = 0, 1, 2, \dots, n$. Then, it is not difficult to see that $w_0 = 3$ and $w_{i+1} = 2w_i + 8$, $i = 1, 2, \dots, n$. This implies that the area of Γ is exponential in the number of layers of G and therefore exponential in the number of vertices of G (recall that G has $n + 1$ layers and $20n + 9$ vertices). $\square$

10.6 Experimental Evaluation

In this section, we present an experimental evaluation of our model. We compare classic orthogonal drawings obtained with the implementation of the original Tamassia algorithm [Tam87] of the yFiles library (http://www.yworks.com) with bend-optimal slog drawings and drawings computed by the heuristic presented in Section 10.3. As a test set, we used the Rome graphs (obtained from http://www.graphdrawing.org) which are approximately 11.500 graphs. We filtered them for connected graphs with maximal degree 4, which left 1.122 graphs. On those we ran our experiments on a linux machine with four cores at $2,5$ GHz and 3 GB of RAM. All implementations were done in Java using the yFiles library. For solving the linear programs to obtain the optimal solutions we used the scip framework [Ach09]. All results we present in this section were computed in less than 1 second each. It is notable that even for graphs with more than 400 vertices the optimal slog drawing could be computed in less than 2 seconds, which suggests that our LP-formulation can be useful for practical applications. To obtain an input for our algorithms we applied the *Smart Organic Layouter* from the yFiles, which is basically a spring embedder algorithm, and as input to our algorithms we used a planarized version of its output. In all following plots the curve denoted by *orth* stands for results for the orthogonal drawings, while the curves denoted by *slog* and *heur* represent the results for bend-optimal and heuristic slog-drawings.

In Figure 10.14 the required area is plotted against the number of vertices. To obtain the actual numbers the results for all graphs with the same number of vertices were averaged. As expected, the area of the slog drawings is larger than that of the orthogonal drawings. A bit surprising in our experiments was that the heuristic tends to require even more area than the optimal solution, although we have the exponential area bound for the optimal case (see Section 10.5). The reason for this is the scaling needed in the heuristic. In the very beginning the orthogonal drawing, from which the heuristic constructs the slog drawing, is scaled up by a factor of five (which yields a factor of 25 in the total area) to gain the space needed for the additional bends. The minimization of the total edge length by the linear program used to obtain the optimal drawing seems to be much more effective than the compaction step in the end of the heuristic algorithm.

As stated in Section 10.2 the number of half-bends in the optimal drawings is at least twice the number of bends in the optimal orthogonal drawing, so in Figure 10.15 we plotted two times the number of orthogonal bends against the number of half-bends produced by our algorithms. Clearly the orthogonal drawings require the least amount of bends. We measured that on average the bend-optimal slog drawings required 4.75 times more half-bends than the orthogonal drawings required bends, while the heuristic drawings required 1.32 times more half-bends than the bend-optimal slog drawings. In actual numbers that means (on average) 13 more bends in the bend-optimal slog drawing and an additional 6 more bends in the drawings produced by the heuristic.

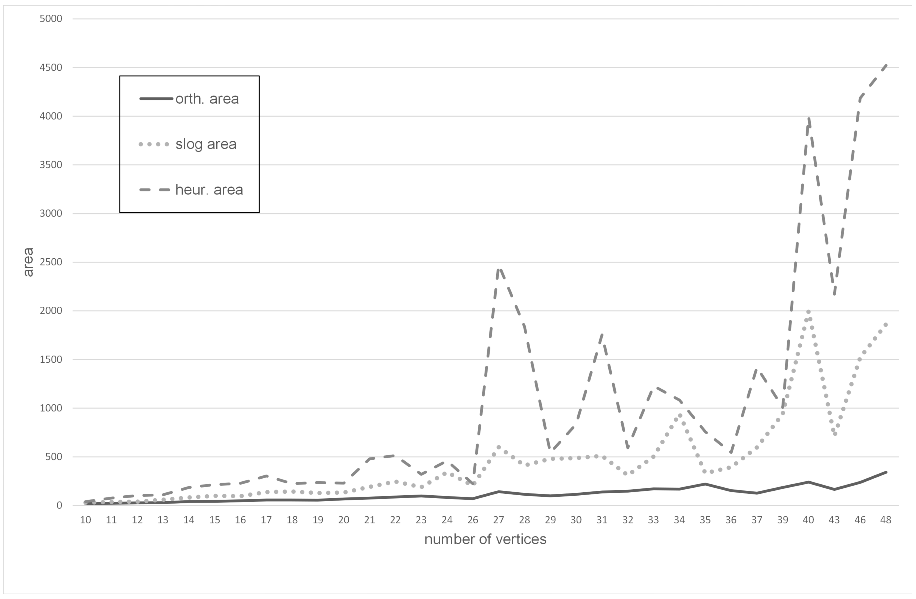


Figure 10.14: Number of vertices against area.

Figure 10.16 shows the total edge length in relation to the number of vertices. In our experiments we found that the plots of the total edge length are comparable to the plots of the area (Figure 10.14). This is exactly as expected, since with larger area the total edge length also has to increase. When comparing the ratio of the longest to the shortest edge, again the orthogonal algorithm produced the smallest results, as can be seen in Figure 10.17. This is because the orthogonal drawings were the most compact ones. For the bend-optimal slog drawing this ratio got up to 39 in our experiments, while the heuristic had an even larger ratio between the longest and the shortest edge of 70 in the largest case we measured. This high ratios are caused by the long diagonal segments required in the slanted model. It seems to us that the heuristic has much higher ratios than the bend-optimal drawings since for the latter we use the LP which minimizes the edge length explicitly, while the heuristic uses no such optimization.

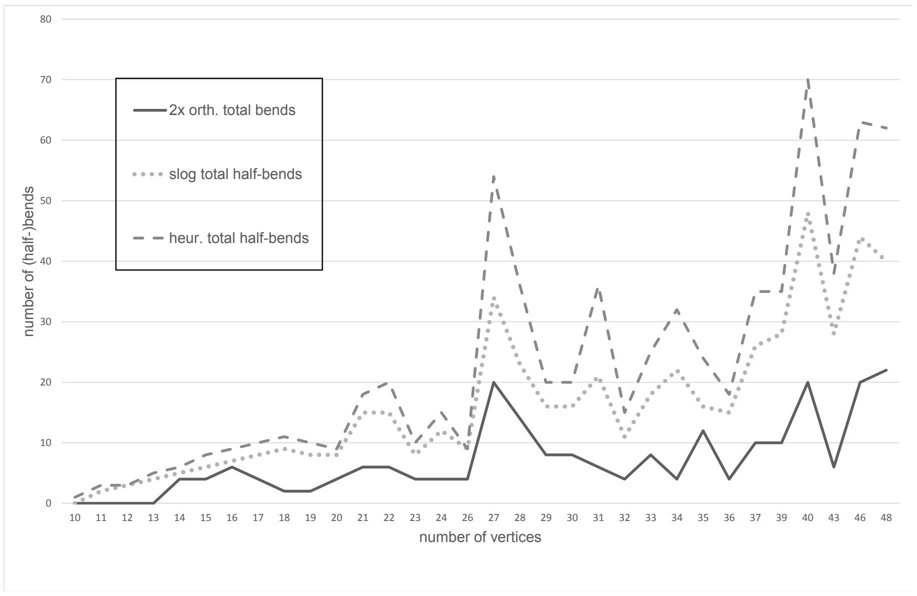


Figure 10.15: Number of vertices against number of bends.

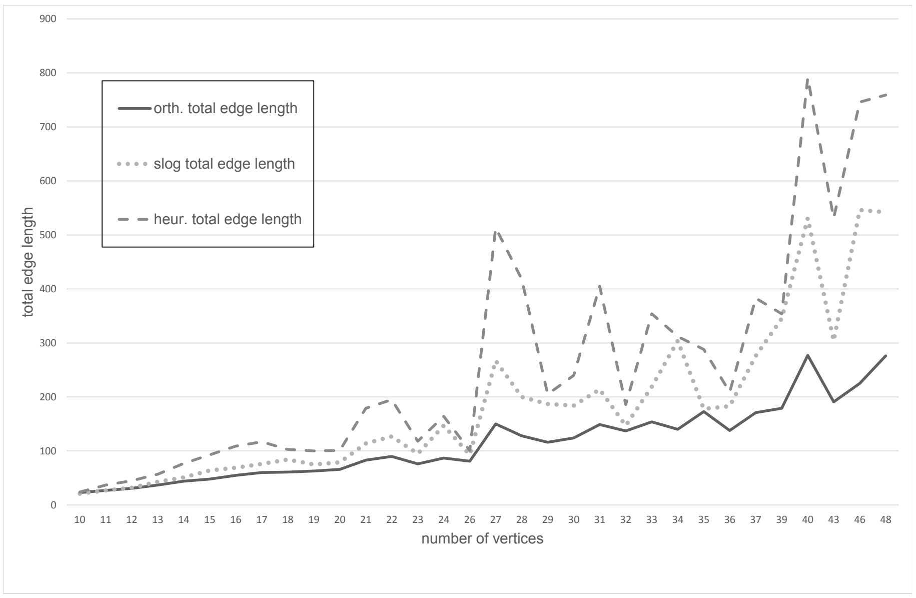


Figure 10.16: Number of vertices against total edge length.

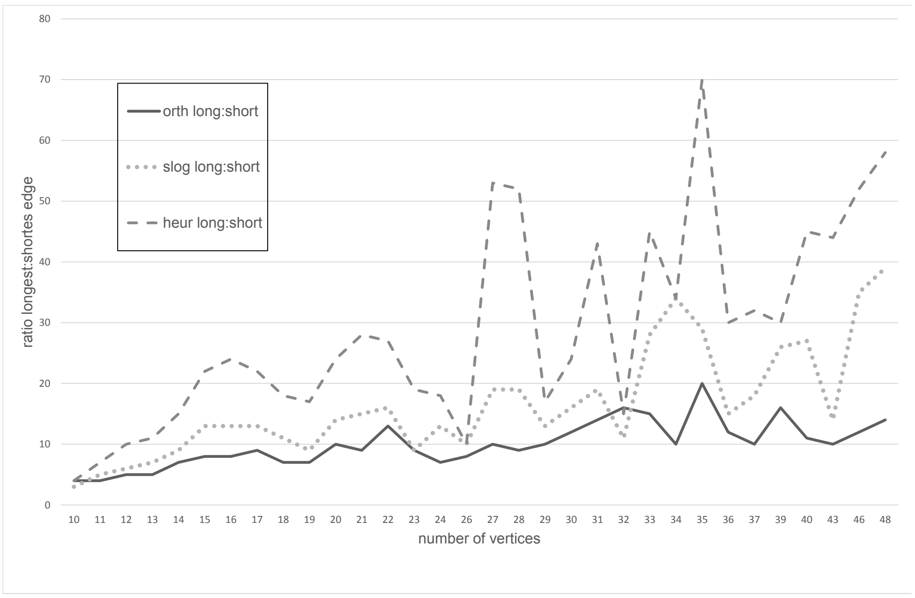


Figure 10.17: Number of vertices against ratio of longest to shortest edge.

10.7 Sample Drawings

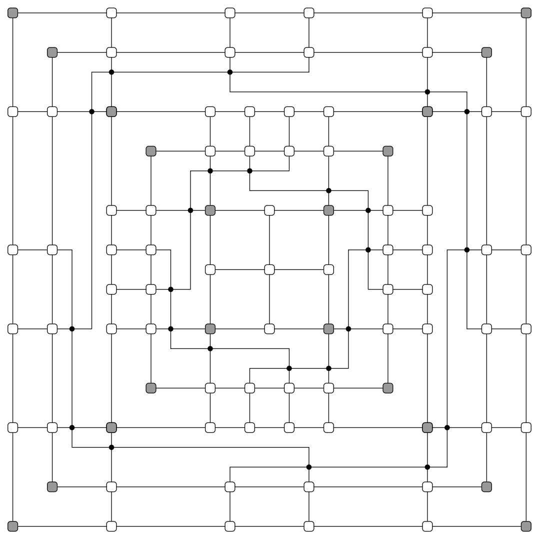


Figure 10.18: An orthogonal drawing of minimum number of bends for the graph of Figure 10.13 establishing the exponential area bound for slog drawings.

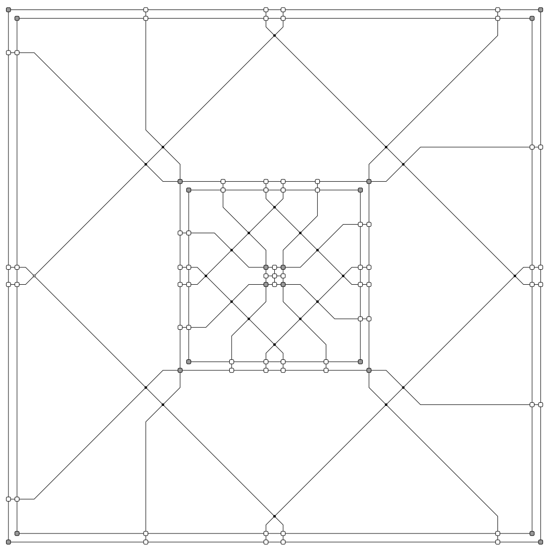


Figure 10.19: The corresponding bend-optimal slog drawing to the one of Figure 10.18.

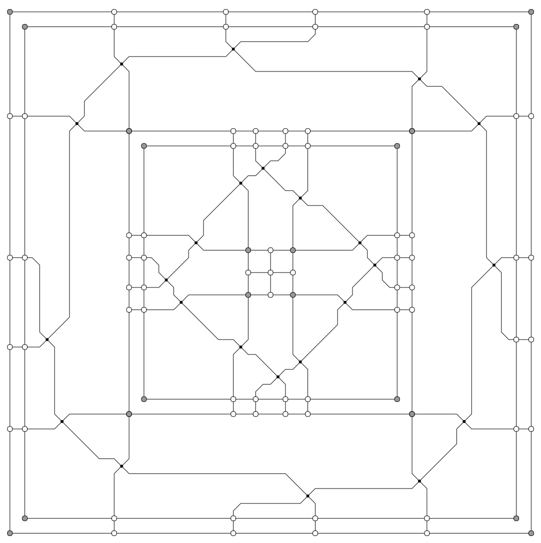


Figure 10.20: The corresponding close-to-optimal slog drawing (to the one of Figure 10.18) produced by our heuristic algorithm of Section 10.3.

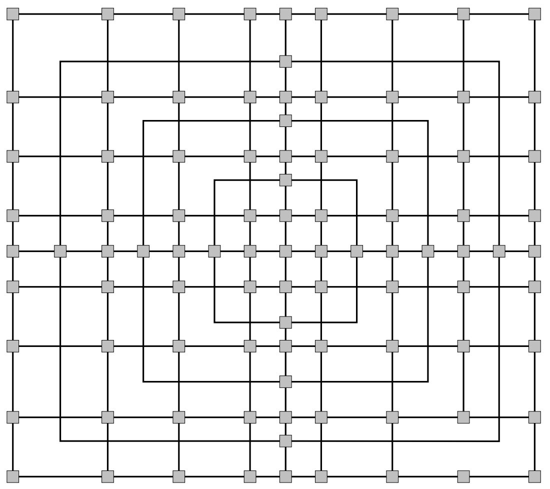


Figure 10.21: A highly symmetric non-planar orthogonal drawing.

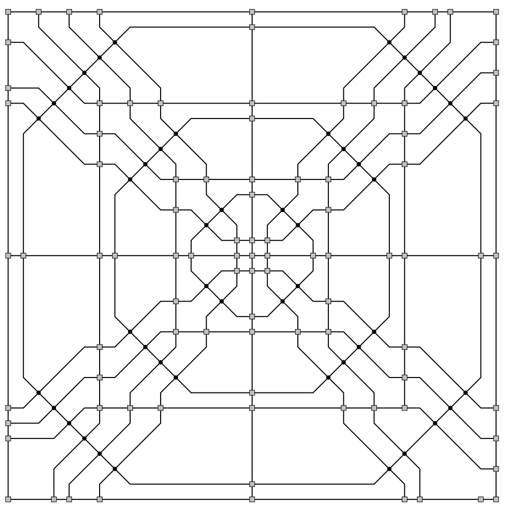


Figure 10.22: The corresponding bend-optimal slog drawing to the one of Figure 10.21.

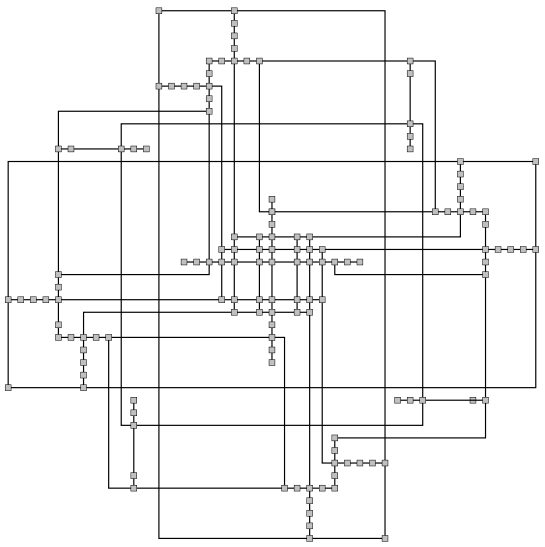


Figure 10.23: A non-planar orthogonal drawing

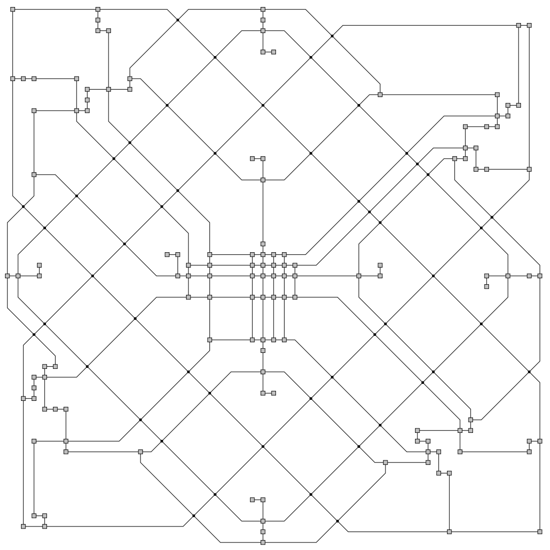


Figure 10.24: The corresponding bend-optimal slog drawing to the one of Figure 10.23.

Chapter 11

Conclusions and Open Problems

In Chapters 9 and 10 we provided algorithms to compute monotone and slanted orthogonal drawings of graph, respectively. In the following, we discuss such results and propose some open problems.

Monotone Drawings of Graphs with Fixed Embedding

In Chapter 9 we studied monotone drawings of graphs in the fixed embedding setting. Since not all embedded planar graphs admit an embedding-preserving monotone drawing with straight-line edges, we focused on computing embedding-preserving monotone drawings with low curve complexity. We proved that curve complexity 2 always suffices and that this bound is worst-case optimal. Furthermore, we described algorithms for computing straight-line monotone drawings for meaningful subfamilies of embedded planar graphs. All the algorithms presented in Chapter 9 can be performed in linear time and most of them produce drawings which require polynomial area.

These results naturally give rise to several interesting open problems; some of them are listed below.

Open Problem 11.1 *Find meaningful subfamilies of embedded planar graphs (other than outerplane graphs and embedded biconnected graphs) that admit monotone drawings with curve complexity smaller than 2.*

Open Problem 11.2 *Is it possible to characterize the embedded planar graphs that admit monotone drawings with curve complexity smaller than 2?*

Open Problem 11.3 *Given an embedded planar graph G_ϕ and an integer $k \in \{0, 1\}$, what is the complexity of deciding whether G_ϕ admits a monotone drawing with curve complexity k?*

Open Problem 11.4 *Given a graph G and an integer $k \in \{0, 1\}$, what is the complexity of deciding whether there exists an embedding ϕ such that G_ϕ admits a monotone drawing with curve complexity k?*

Open Problem 11.5 *Given a graph G and an integer $k \in \{0, 1\}$, what is the complexity of deciding whether there exists an embedding ϕ such that G_ϕ does not admit any monotone drawing with curve complexity k?*

Notice that, although Problems 11.3-11.5 are related, there is no evidence that answering one of them implies an answer for any other.

Observe that the length of the edges and the angles β_{μ_i} and α_{μ_i} in a drawing produced by the algorithm described in Theorem 9.3 are reduced at each step, which implies that the area of the drawing is not polynomially bounded.

Open Problem 11.6 *Is there any algorithm that computes monotone drawings of embedded biconnected planar graphs in polynomial area?*

Open Problem 11.7 *Is there any algorithm that computes monotone drawings of outerplane graphs in subcubic area?*

Another problem, related both to morphing and monotone drawings, is formulated in the following.

Open Problem 11.8 *Given any two monotone drawings Γ_s and Γ_t of a plane graph G, does there exists a morph such that at each time instant the drawing of G is monotone? If so, does it require additional bends?*

Slanted Orthogonal Drawings

We introduced a new model for drawing graphs of max-degree four, in which orthogonal bends are replaced by pairs of "slanted" bends and crossings occur on diagonal segments only. The main advantage of this model is that, even in drawings of large graphs (where vertices might not be clearly visible), it is immediately clear which pair of edges induce a crossing and where such a crossing is located in the drawing. We presented an algorithm to construct slog drawings with almost-optimal number of bends and quadratic area, for general max-degree four graphs. By a modification of Tamassia's min-cost flow approach, we showed that a bend-optimal representation of the graph can efficiently be computed in polynomial time and we presented an LP-approach to compute a corresponding drawing.

Open Problem 11.9 *Does every max-degree four graph admit such a bend-optimal drawing?*

Our experiments led us to believe that it is possible, although we could not prove it.

Variants of our basic model may lead to even more flexibility for the drawings. An extension to support higher degree graphs will be necessary to make the approach practical.

Another interesting problem concerns the morph of slog drawings.

Open Problem 11.10 *Given any two slog drawing Γ_s and Γ_t of the same graph such that crossings appear between the same pairs of edges in both drawings, does there exists a morph transforming Γ_s into Γ_t such that the drawing resulting at the end of each step is a slog drawing?*

Appendices

Appendix A: 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:

Point-set Embedding of Graphs.

A *planar straight-line embedding* of a graph G *into a point set* P is a mapping of each vertex of G to a distinct point of P and of each edge of G to the straight-line segment between the corresponding endpoints so that no two edges cross. Let $\mathcal{G}$ be a class of n-vertex planar graphs and P be a point set of size m, with $m \geq n$. Point set P is *universal* for the class $\mathcal{G}$ if for every $G \in \mathcal{G}$, G has a planar straight-line embedding into P.

Asymptotically, the smallest universal point set for general planar graphs is known to have size at least $1.235n$ [CK89, Kur04], while the best known upper bound is $O(n^2)$ [CN98, dPP90, Sch90]. Characterizing the asymptotic size of the smallest universal point set is a well-known open problem also referred in [OPG, Cab06, DMO].

A subclass of planar graphs for which a "small" universal point set is known is the class of outerplanar graphs, that is, the graphs that admit a straight-line planar embedding with all vertices incident to the outer face. Gritzmann *et al.* [GMPP91] and Bose [Bos02] proved that any point set of size n is universal for outerplanar graphs. In [GMPP91] it is noticed that outerplanar graphs are the largest class of graphs for which any arbitrary point set is universal.

A generalization of outerplanar graphs are k-outerplanar graphs, $k \geq 2$. A planar embedding of a graph is *k-outerplanar* if removing the vertices of the outer face yields a $(k-1)$-outerplanar embedding, where 1-outerplanar is an outerplanar embedding. Vertices removed at the i-th step are at level i. A graph is *k-outerplanar* if it admits a

k-outerplanar embedding. Note that no (arbitrarily large) convex point set is universal for k-outerplanar graphs, $k \geq 2$.

The decision question of whether a given planar graph admits a planar straight-line embedding into a given point set of the same size was proved to be NP-hard, even for 2-outerplanar graphs and 3-level point sets [Cab06].

A k-outerplanar graph is *simply-nested* [Cim90] if levels 1 to $k-1$ are chordless cycles and level k is either a cycle or a tree. A planar graph is *simply-nested* if it is k-outerplanar simply-nested for some $k \leq n$. Simply-nested graphs turned out to be useful to derive some properties of planar graphs. Cimikowski [Cim90] proved hamiltonicity of simply-nested planar triangulations. Baker [Bak94] used these graphs to derive approximation algorithms for various NP-complete problems on planar graphs. A variant of nested triangulations was explored by Yannakakis in his celebrated result on book embeddings of planar graphs [Yan89].

We show a $O(n(\frac{\log n}{\log \log n})^2)$-size universal point set for simply-nested n-vertex graphs. Such result is based on the construction of a $8n + 8$-size universal point set for simply-nested n-vertex graphs for which the number of vertices on each of level is known in advance.

Our results find applications to another class of graphs, quite popular in Graph Drawing. In [BBF05] Bachmaier *et al.* defined a graph to be (*proper*) *k-radial planar* if given a partition of its vertices into k concentric circles, its edges can be drawn as monotonic curves between (consecutive) circles without crossings and showed that radial planarity is decidable in linear time. Our results give a small universal point set for proper k-radial planar graphs, since they can be easily proved to be a subclass of simply-nested planar graphs.

Clustered Planarity.

Clustered planarity is a classical Graph Drawing topic (see [CD05] for a survey). A *clustered graph* $C(G,T)$ consists of a graph G and of a rooted tree T whose leaves are the vertices of G. Such a structure is used to enrich the vertices of the graph with hierarchical information. In fact, each internal node μ of T represents the subset, called *cluster*, of the vertices of G that are the leaves of the subtree of T rooted at μ. Tree T, which defines the inclusion relationships among clusters, is called *inclusion tree*, while G is the *underlying graph* of $C(G,T)$.

In a *drawing* of a clustered graph $C(G,T)$ vertices and edges of G are drawn as points and open curves, respectively, and each node μ of T is represented by a simple closed region $R(\mu)$ containing exactly the vertices of μ. Also, if μ is a descendant of a node ν, then $R(\nu)$ contains $R(\mu)$.

A drawing of C can have three types of crossings. *Edge-edge crossings* are crossings between edges of G. Algorithms to produce drawings allowing edge-edge crossings have already been proposed (see, for example, [DDM02] and Figure A.1(a)). Two kinds of crossings involve regions, instead. Consider an edge e of G and a node μ of T. If e intersects the boundary of $R(\mu)$ only once, this is not considered as a crossing since there is no way of connecting the endpoints of e without intersecting the boundary of $R(\mu)$. On the contrary, if e intersects the boundary of $R(\mu)$ more than once, we have *edge-region crossings*. An example of this kind of crossings is provided by Figure A.1(b), where edge (u, w) traverses $R(\mu)$ and edge (u, v) exits and enters $R(\mu)$. Finally, consider two nodes μ and ν of T; if the boundary of $R(\mu)$ intersects the boundary of $R(\nu)$, we have a *region-region crossing* (see Figure A.1(c) for an example).

A drawing of a clustered graph is *c-planar* if it does not have any edge-edge, edge-region, or region-region crossing. A clustered graph is *c-planar* if it admits a c-planar drawing.

In the last decades c-planarity has been deeply studied. While the complexity of deciding if a clustered graph is c-planar is still an open problem in the general case, polynomial-time algorithms have been proposed to test c-planarity and produce c-planar drawings under several kinds of restrictions, such as:

- Assuming that each cluster induces a small number of connected components ([CW06, CDF$^+$08, Dah98, FCE95b, FCE95a, GLS05, GJL$^+$02, JJKL08, JSTV08]). In particular, the case in which the graph is *c-connected*, that is, for each node ν of T the graph induced by the vertices of ν is connected, has been deeply investigated.

- Considering only *flat* hierarchies, i.e., the height of T is two, namely no cluster different from the root contains other clusters ([CDPP05, CBPP09, DF09]).

- Focusing on particular families of underlying graphs ([CDPP05, CBPP09, JKK$^+$07]).

- Fixing the embedding of the underlying graph ([DF09, JJKL08]).

This huge body of research can be read as a collection of polynomial-time testable sufficient conditions for c-planarity.

In contrast, the planarity of the underlying graph is the only polynomial-time testable necessary condition that has been found so far for c-planarity in the general case. Such a condition, however, is not sufficient and the consequences on the problem due to the requirement of not having edge-region and region-region crossings are not yet fully understood.

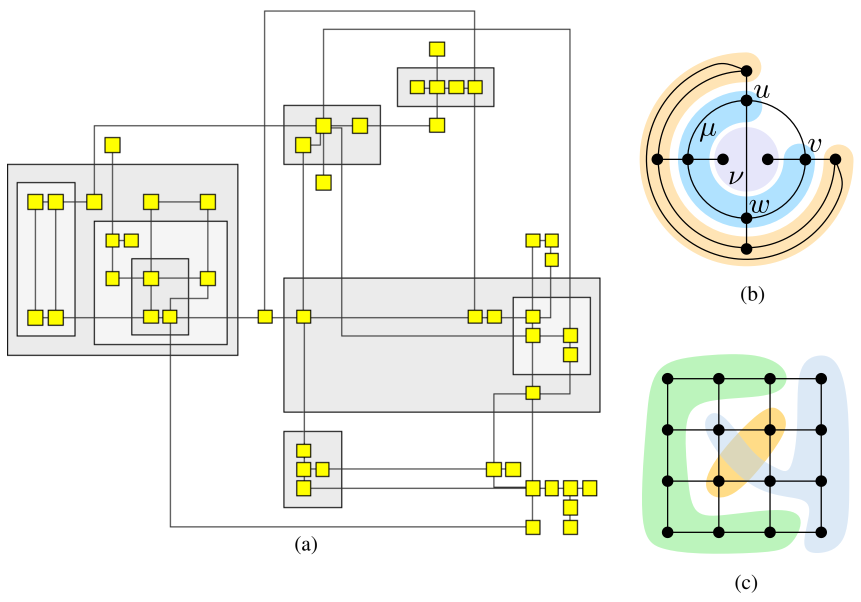


Figure A.1: Examples of crossings in drawings of clustered graphs. (a) A drawing obtained with the planarization algorithm described in [DDM02] and containing three edge-edge crossings. (b) A drawing with two edge-region crossings. (c) A drawing with a region-region crossing.

Other known necessary conditions are either trivial (i.e., satisfied by all clustered graphs) or of unknown complexity as the original problem is. An example of the first kind is the existence of a c-planar clustered graph obtained by splitting some cluster into sibling clusters [AFP09]. An example of the second kind, which is also a sufficient condition, is the existence of a set of edges that, if added to the underlying graph, make the clustered graph c-connected and c-planar [FCE95b].

In our paper we study a relaxed model of c-planarity. Namely, we study $\langle\alpha, \beta, \gamma\rangle$-drawings of clustered graphs. In an $\langle\alpha, \beta, \gamma\rangle$-drawing the number of edge-edge, edge-region, and region-region crossings is equal to α, β, and γ, respectively. Figure A.1 shows examples of a $\langle 3, 0, 0\rangle$-drawing, a $\langle 0, 2, 0\rangle$-drawing, and a $\langle 0, 0, 1\rangle$-drawing, respectively. Notice that this model provides a generalization of c-planarity, as the traditional c-planar drawing is a special case of an $\langle\alpha, \beta, \gamma\rangle$-drawing where $\alpha = \beta = \gamma = 0$. Hence, we can say that the existence of an $\langle\alpha, \beta, \gamma\rangle$-drawing, for some values

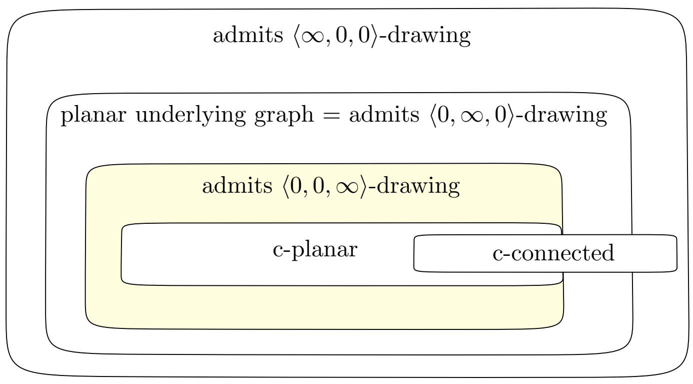

admits $\langle\infty, 0, 0\rangle$-drawing
planar underlying graph = admits $\langle 0, \infty, 0\rangle$-drawing
admits $\langle 0, 0, \infty\rangle$-drawing
c-planar
c-connected

Figure A.2: Containment relationships among instances of clustered planarity. The existence of a $\langle 0, 0, \infty\rangle$-drawing is a necessary condition for c-planarity.

of α, β, and γ, is a necessary condition for c-planarity.

In our study we focus on clustered graphs whose underlying graph is planar. We mainly concentrate on the existence of drawings in which only one type of crossings is allowed. We call these drawings $\langle\infty, 0, 0\rangle$-, $\langle 0, \infty, 0\rangle$-, and $\langle 0, 0, \infty\rangle$-drawings, respectively. Our investigation uncovers that allowing different types of crossings has a different impact on the existence of drawings of clustered graphs (see Figure A.2). In particular, we prove that, while every clustered graph admits an $\langle\infty, 0, 0\rangle$-drawing (even if its underlying graph is not planar) and a $\langle 0, \infty, 0\rangle$-drawing, there exist clustered graphs not admitting any $\langle 0, 0, \infty\rangle$-drawing. Further, we provide a polynomial-time testing algorithm to decide whether a biconnected clustered graph admits a $\langle 0, 0, \infty\rangle$-drawing. From this fact we conclude that the existence of such a drawing is the first non-trivial necessary condition for the c-planarity of clustered graphs that can be tested efficiently. This allows us to further restrict the search for c-planar instances with respect to the obvious condition that the underlying graph is planar.

Also, we investigate the relationships among the minimum number of edge-edge, edge-region, and region-region crossings for drawings of the same clustered graph, showing that, in most of the cases, the fact that a clustered graph admits a drawing with few crossings of one type does not imply that such a clustered graph admits a drawing with few crossings of another type.

Finally, we show that minimizing the sum $\alpha + \beta + \gamma$ in a $\langle\alpha, \beta, \gamma\rangle$-drawing of a clustered graph is an NP-complete problem. Since in our construction it is possible to replace each crossing of any type with a crossing of a different type, this implies that the problems of minimizing crossings in $\langle \infty, 0, 0 \rangle$-, $\langle 0, \infty, 0 \rangle$-, and $\langle 0, 0, \infty \rangle$-drawings are also NP-complete. However, for the first two types of drawings we can prove NP-completeness even for simpler classes of clustered graphs.

We remark that drawings of clustered graphs where a few intersections are admitted may meet the requirements of many typical Graph Drawing applications, and that their employment is encouraged by the fact that the class of c-planar instances might be too small to be relevant for some application contexts.

More in detail, we present the following results (recall that we assume the necessary condition that the underlying graph is planar to be always satisfied):

(i) We provide algorithms to produce $\langle \infty, 0, 0 \rangle$-, $\langle 0, \infty, 0 \rangle$-, and $\langle 0, 0, \infty \rangle$-drawings of clustered graphs, if they exist. In particular, while $\langle \infty, 0, 0 \rangle$- and $\langle 0, \infty, 0 \rangle$-drawings always exist, we show that some clustered graphs do not admit any $\langle 0, 0, \infty \rangle$-drawing, and we present a polynomial-time algorithm to test whether a biconnected clustered graph admits a $\langle 0, 0, \infty \rangle$-drawing, which is a necessary condition for c-planarity. The algorithm, whose approach is reminiscent of [ABF$^+$12], is based on a characterization of the planar embeddings that lead to $\langle 0, 0, \infty \rangle$-drawings, and on a subsequent structural characterization of the existence of a $\langle 0, 0, \infty \rangle$-drawing for any biconnected clustered graph $C(G, T)$, based on the SPQR-tree decomposition of G.

(ii) The above mentioned algorithms provide upper bounds on the number of crossings for the three kinds of drawings. We show that the majority of these upper bounds are tight by providing matching lower bounds. These results are summarized in Table A.1.

(iii) We show that there are clustered graphs admitting drawings with one crossing of a certain type but requiring many crossings in drawings where only different types of crossings are allowed. For example, there are clustered graphs that admit a $\langle 1, 0, 0 \rangle$-drawing and that require $\beta \in \Omega(n^2)$ in any $\langle 0, \beta, 0 \rangle$-drawing and $\gamma \in \Omega(n^2)$ in any $\langle 0, 0, \gamma \rangle$-drawing. See Table A.2 for a summary of these results.

(iv) We present several complexity results. Namely, we show that:

- minimizing $\alpha + \beta + \gamma$ in an $\langle \alpha, \beta, \gamma \rangle$-drawing is NP-complete even if the underlying graph is planar, namely a forest of star graphs;
- minimizing α in an $\langle \alpha, 0, 0 \rangle$-drawing is NP-complete even if the underlying graph is a matching;

c-c	flat	$\langle \alpha, 0, 0 \rangle$		$\langle 0, \beta, 0 \rangle$		$\langle 0, 0, \gamma \rangle$	
		α UB	α LB	β UB	β LB	γ UB	γ LB
NO	NO	$O(n^2)$	$\Omega(n^2)$	$O(n^3)$	$\Omega(n^2)$	$O(n^3)^{✠}$	$\Omega(n^3)$
NO	YES	$O(n^2)$	$\Omega(n^2)$	$O(n^2)$	$\Omega(n^2)$	$O(n^2)^{✠}$	$\Omega(n^2)$
YES	NO	$O(n^2)$	$\Omega(n^2)$	$O(n^2)$	$\Omega(n^2)$	$0^{✠}$ [FCE95b]	$0^{✠}$ [FCE95b]
YES	YES	$O(n^2)$	$\Omega(n^2)$	$O(n)$	$\Omega(n)$	$0^{✠}$ [FCE95b]	$0^{✠}$ [FCE95b]

Table A.1: Upper and lower bounds for the number of crossings in $\langle \infty, 0, 0 \rangle$-, $\langle 0, \infty, 0 \rangle$-, and $\langle 0, 0, \infty \rangle$-drawings of clustered graphs. Flags *c-c* and *flat* mean that the clustered graph is *c-connected* and that the cluster hierarchy is *flat*, respectively. Results written in gray derive from those in black (proved in [ADD$^+$12]), while a “✠” means that there exist clustered graphs not admitting the corresponding drawings. A “0” occurs if the clustered graph is c-planar.

$\rightarrow$	$\langle \alpha, 0, 0 \rangle$	$\langle 0, \beta, 0 \rangle$	$\langle 0, 0, \gamma \rangle$
$\langle 1, 0, 0 \rangle$	▨	$\Omega(n^2)$	$\Omega(n^2)$
$\langle 0, 1, 0 \rangle$	$\Omega(n)$	▨	$\Omega(n^2)$
$\langle 0, 0, 1 \rangle$	$\Omega(n^2)$	$\Omega(n)$	▨

Table A.2: Relationships between types of drawings proved in [ADD$^+$12].

- minimizing β in a $\langle 0, \beta, 0 \rangle$-drawing is NP-complete (see also [For05]) even for c-connected flat clustered graphs in which the underlying graph is a triconnected planar multigraph;

Appendix B: List of Publications

Journal Publications

- Patrizio Angelini, Walter Didimo, Stephen G. Kobourov, Tamara Mchedlidze, Vincenzo Roselli, Antonios Symvonis, and Stephen K. Wismath. Monotone drawings of graphs with fixed embedding. *Algorithmica*, pages 1–25, 2013.

Conference Publications

- Michael A. Bekos, Michael Kaufmann, Robert Krug, Stefan Naher, and Vincenzo Roselli. Slanted orthogonal drawings. In Stephen Wismath, Alexander Wolff, Stephen Wismath, and Alexander Wolff, editors, *21st International Symposium on Graph Drawing (GD '13)*, volume 8242 of *Lecture Notes in Computer Science*, pages 428–439, 2013.

- Patrizio Angelini, Giordano Da Lozzo, GiuseppeDi Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Morphing planar graph drawings optimally. In *Proceedings of 41st International Colloquium on Automata, Languages and Programming (ICALP '14)*, 2014. To appear.

- Patrizio Angelini, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Morphing planar graph drawings efficiently. In Stephen Wismath and Alexander Wolff, editors, *21st International Symposium on Graph Drawing (GD '13)*, volume 8242 of *Lecture Notes in Computer Science*, pages 49–60, 2013.

- Soroush Alamdari, Patrizio Angelini, Timothy M. Chan, Giuseppe Di Battista, Fabrizio Frati, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan T. Wilkinson. Morphing planar graph drawings with a polynomial number of steps. In Sanjeev Khanna, editor, *Proceedings of the*

Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 1656–1667. SIAM, 2013.

- Patrizio Angelini, Walter Didimo, Stephen Kobourov, Tamara Mchedlidze, Vincenzo Roselli, Antonios Symvonis, and Stephen Wismath. Monotone drawings of graphs with fixed embedding. In *19th International Symposium on Graph Drawing (GD '11)*, Lecture Notes in Computer Science, pages 379–390, 2011.

- Patrizio Angelini, Giuseppe Di Battista, Michael Kaufmann, Tamara Mchedlidze, Vincenzo Roselli, and Claudio Squarcella. Small point sets for simply-nested planar graphs. In *19th International Symposium on Graph Drawing (GD '11)*, Lecture Notes in Computer Science, pages 75–85, 2011.

Technical Reports

- Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, Vincenzo Roselli. Morphing Planar Graph Drawings Optimally. Technical Report arXiv:1402.4364, Cornell University, 2014.

- Patrizio Angelini, Fabrizio Frati, Maurizio Patrignani, Vincenzo Roselli. Morphing Planar Graph Drawings Efficiently. Technical Report arXiv:1308.4291, Cornell University, 2013.

- Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, Vincenzo Roselli. Relaxing the Constraints of Clustered Planarity . Technical Report arXiv:1207.3934, Cornell University, 2012.

Others

- Vincenzo Roselli. Animazione di grafi: Morphing di strutture planari. Master's thesis, Roma Tre University, Rome, Italy, 2010.

Bibliography

[AAC$^+$13] Soroush Alamdari, Patrizio Angelini, Timothy M. Chan, Giuseppe Di Battista, Fabrizio Frati, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan T. Wilkinson. Morphing planar graph drawings with a polynomial number of steps. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1656–1667. SIAM, 2013. 29, 51, 77

[AAD$^+$11] Oswin Aichholzer, Greg Aloupis, Erik D. Demaine, Martin L. Demaine, Vida Dujmovic, Ferran Hurtado, Anna Lubiw, Günter Rote, André Schulz, Diane L. Souvaine, and Andrew Winslow. Convexifying polygons without losing visibilities. In Greg Aloupis and David Bremner, editors, *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry*, pages 229–334, 2011. 28

[ABF$^+$12] Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. Testing the simultaneous embeddability of two graphs whose intersection is a biconnected or a connected graph. *Journal of Discrete Algorithms*, 14:150–172, 2012. Selected papers from the 21st International Workshop on Combinatorial Algorithms (IWOCA 2010). 168

[ACB$^+$12] Patrizio Angelini, Enrico Colasante, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani. Monotone drawings of graphs. *Journal of Graph Algorithms and Applications*, 16(1):5–35, 2012. Special Issue on Selected Papers from GD '10. 100, 101, 102, 104, 107, 114, 115, 116, 121, 122

[ACD$^+$12] Patrizio Angelini, Enrico Colasante, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani. Monotone drawings of graphs. *Jour-*

nal of Graph Algorithms and Applications, 16(1):5–35, 2012. Special Issue from GD '10. 23

[ACDP08] Patrizio Angelini, Pier Francesco Cortese, Giuseppe Di Battista, and Maurizio Patrignani. Topological morphing of planar graphs. In Ioannis G. Tollis and Maurizio Patrignani, editors, *International Symposium on Graph Drawing (GD '08)*, volume 5417 of *LNCS*, pages 145–156, 2008. 29

[ACDP13] Patrizio Angelini, Pier Francesco Cortese, Giuseppe Di Battista, and Maurizio Patrignani. Topological morphing of planar graphs. *Theoretical Computer Science*, 514:2–20, 2013. 29

[Ach09] Tobias Achterberg. Scip: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, July 2009. `http://mpc.zib.de/index.php/MPC/article/view/4`. 147

[ACM89] Esther M. Arkin, Robert Connelly, and Joseph S. B. Mitchell. On monotone paths among obstacles with applications to planning assemblies. In Kurt Mehlhorn, editor, *Symposium on Computational Geometry*, pages 334–343, 1989. 100, 121

[ADD$^+$12] Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Relaxing the constraints of clustered planarity. Technical Report arXiv:1207.3934, Cornell University, 2012. 169

[ADD$^+$14] Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Morphing planar graph drawings optimally. In *Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP '14)*, 2014. To appear. 29, 95

[ADK$^+$11] Patrizio Angelini, Walter Didimo, Stephen Kobourov, Tamara Mchedlidze, Vincenzo Roselli, Antonios Symvonis, and Stephen Wismath. Monotone drawings of graphs with fixed embedding. In Marc van Kreveld and Bettina Speckmann, editors, *19th International Symposium on Graph Drawing (GD '11)*, Lecture Notes in Computer Science, pages 379–390, 2011. 99

[ADK$^+$13] Patrizio Angelini, Walter Didimo, Stephen G. Kobourov, Tamara Mchedlidze, Vincenzo Roselli, Antonios Symvonis, and Stephen K.

Wismath. Monotone drawings of graphs with fixed embedding. *Algorithmica*, pages 1–25, 2013. 23, 99

[AFG10] Patrizio Angelini, Fabrizio Frati, and Luca Grilli. An algorithm to construct greedy drawings of triangulations. *Journal of Graph Algorithms and Applications*, 14(1):19–51, 2010. 100

[AFP09] Patrizio Angelini, Fabrizio Frati, and Maurizio Patrignani. Splitting clusters to get c-planarity. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing*, volume 5849 of *Lecture Notes in Computer Science*, pages 57–68. Springer, 2009. 166

[AFPR13] Patrizio Angelini, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Morphing planar graph drawings efficiently. In Stephen Wismath and Alexander Wolff, editors, *21st International Symposium on Graph Drawing (GD '13)*, volume 8242 of *Lecture Notes in Computer Science*, pages 49–60, 2013. 42, 51, 57

[AHU83] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983. 7

[Ale23] James W. Alexander. On the deformation of an n cell. *Proceedings of the National Academy of Sciences*, 9(12):406–407, 1923. 27

[AP89] Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, 1989. 14

[Bak94] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41:153–180, 1994. 164

[BBF05] Christian Bachmaier, Franz J. Brandenburg, and Michael Forster. Radial level planarity testing and embedding in linear time. *Journal of Graph Algorithms and Applications*, 9(1):53–97, 2005. 164

[BFM07] Nicolas Bonichon, Stefan Felsner, and Mohamed Mosbah. Convex drawings of 3-connected plane graphs. *Algorithmica*, 47(4):399–420, 2007. 17

[BHL13] Fidel Barrera-Cruz, Penny Haxell, and Anna Lubiw. Morphing planar graph drawings with unidirectional moves. Mexican Conference on Discrete Mathematics and Computational Geometry, 2013. 29, 50, 51, 52, 54, 55

[BK94] Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. In Jan van Leeuwen, editor, *Proceedings of 2nd European Symposium on Algorithms*, volume 855 of *LNCS*, pages 24–35, 1994. 127

[BKK$^+$13] Michael A. Bekos, Michael Kaufmann, Robert Krug, Stefan Naher, and Vincenzo Roselli. Slanted orthogonal drawings. In Stephen Wismath and Alexander Wolff, editors, *21st International Symposium on Graph Drawing (GD '13)*, volume 8242 of *Lecture Notes in Computer Science*, pages 428–439, 2013. 125

[BKKS12] Michael A. Bekos, Michael Kaufmann, Stephen G. Kobourov, and Antonios Symvonis. Smooth orthogonal layouts. In Walter Didimo and Maurizio Patrignani, editors, *Proceedings of 20th International Symposium on Graph Drawing*, volume 7704 of *LNCS*, pages 150–161, 2012. 128

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property interval graphs and graph planarity using PQ-tree algorithms. *Journal of Computer and System*, 13:335–379, 1976. 12

[BLPS13] Therese Biedl, Anna Lubiw, Mark Petrick, and Michael Spriggs. Morphing orthogonal planar graph drawings. *ACM Transactions on Algorithms*, 9(4):29:1–29:24, October 2013. 28

[BLS05] Therese C. Biedl, Anna Lubiw, and Michael J. Spriggs. Morphing planar graphs while preserving edge directions. In Patrick Healy and Nikola S. Nikolov, editors, *Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2005. 28

[BM76] John A. Bondy and Uppaluri S. R Murty. *Graph theory with applications*. American Elsevier Publishing Co., Inc., New York, 1976. 7

[BM04] John M. Boyer and Wendy J. Myrvold. On the cutting edge: simplified $O(n)$ planarity by edge addition. *Journal of Graph Algorithms and Applications*, 8(3):241–273, 2004. 12

[Bos02] Prosenjit Bose. On embedding an outer-planar graph in a point set. *Computational Geometry: Theory and Applications*, 23(3):303–312, November 2002. 163

[Bro60] Achille Brocot. Calcul des rouages par approximation, nouvelle methode. *Revue Chronometrique*, 6:186–194, 1860. 101

[BS78a] R. H. Bing and Michael Starbird. Linear isotopies in E^2. *Transactions of the American Mathematical Society*, 237:205–222, 1978. 28

[BS78b] R. H. Bing and Michael Starbird. Super triangulations. *Pacific Journal of Mathematics*, 74(2):307–325, 1978. 28

[BSW97] Mark Babikov, Diane L. Souvaine, and Rephael Wenger. Constructing piecewise linear homeomorphisms of polygons with holes. In David Rappaport, editor, *Proceedings of the 9th Canadian Conference on Computational Geometry*, 1997. 28

[Cab06] Sergio Cabello. Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. *Journal of Graph Algorithms and Applications*, 10(2):353–363, 2006. 163, 164

[Cai44a] Steward S. Cairns. Deformations of plane rectilinear complexes. *American Mathematical Monthly*, 51(5):247–252, 1944. 27, 30, 80, 83

[Cai44b] Stewart S. Cairns. Isotopic deformations of geodesic complexes on the 2-sphere and on the plane. *Annals of Mathematics*, 45(2):207–217, 1944. 28

[CBPP09] Pier Francesco Cortese, Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia. On embedding a cycle in a plane graph. *Discrete Mathematics*, 309(7):1856–1869, 2009. 165

[CD05] Pier Francesco Cortese and Giuseppe Di Battista. Clustered planarity (invited lecture). In Joe Mitchell and Günter Rote, editors, *Twenty-first annual symposium on Computational Geometry (proc. SoCG 05)*, ACM, pages 30–32, 2005. 164

[CDF$^+$08] Pier Francesco Cortese, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Maurizio Pizzonia. C-planarity of c-connected clustered graphs. *Journal of Graph Algorithms and Applications*, 12(2):225–262, 2008. 165

[CDPP05] Pier Francesco Cortese, Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia. On embedding a cycle in a plane graph. In Patrick Healy and Nikola S. Nikolov, editors, *Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 49–60. Springer, 2005. 165

[CdVPV03] Éric Colin de Verdière, Michel Pocchiola, and Gert Vegter. Tutte’s barycenter method applied to isotopies. *Computational Geometry*, 26(1):81 – 97, 2003. 29

[CGC+02] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. A multiresolution framework for dynamic deformations. In John F. Hughes, editor, *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA ’02, pages 41–47, New York, NY, USA, 2002. ACM. 29

[Cha91] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(5):485–524, 1991. 43, 44, 81, 82, 108

[Cim90] Robert J. Cimikowski. Finding hamiltonian cycles in certain planar graphs. *Information Processing Letters*, 35(5):249 – 254, 1990. 164

[CK89] Marek Chrobak and Howard J. Karloff. A lower bound on the size of universal sets for planar graphs. *SIGACT News*, 20(4):83–86, 1989. 163

[CK12] Sabine Cornelsen and Andreas Karrenbauer. Accelerated bend minimization. *Journal of Graph Algorithms and Applications*, 16(3):635–650, 2012. 127

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009. 7

[CN88] Norishige Chiba and Takao Nishizeki. *Planar Graphs: Theory and Algorithms*. Annals of Discrete Mathematics 32. North-Holland, Amsterdam, 1988. 7, 121

[CN98] Marek Chrobak and Shin-Ichi Nakano. Minimum-width grid drawings of plane graphs. *Computational Geometry*, 11(1):29 – 54, 1998. 17, 163

[CON85] Norishige Chiba, Kazunori Onoguchi, and Takao Nishizeki. Drawing plane graphs nicely. *Acta Informatica*, 22:187–201, 1985. 121

[CR05] Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 34(4):825–847, 2005. 14

[CW06] Sabine Cornelsen and Dorothea Wagner. Completely connected clustered graphs. *Journal of Discrete Algorithms*, 4(2):313–323, 2006. 165

[Dah98] Elias Dahlhaus. A linear time algorithm to recognize clustered graphs and its parallelization. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *Latin American Theoretical Informatics (LATIN '98)*, LNCS, pages 239–248, 1998. 165

[dCvO08] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd edition, 2008. 7

[DDM02] Giuseppe Di Battista, Walter Didimo, and Alessandro Marcandalli. Planarization of clustered graphs. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 60–74, 2002. 165, 166

[DETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999. 7, 100

[DF09] Giuseppe Di Battista and Fabrizio Frati. Efficient c-planarity testing for embedded flat clustered graphs with small faces. *Journal of Graph Algorithms and Applications*, 13(3):349–378, 2009. 165

[dFOdM12] Hubert de Fraysseix and Patrice Ossona de Mendez. Trémaux trees and planarity. *European Journal of Combinatorics*, 33(3):279–293, April 2012. 12

[Die05] Reinhard Diestel. *Graph theory*. Graduate texts in mathematics. Springer, Berlin, 2005. 7

[DMO] Erik D. Demaine, Joseph S. B. Mitchell, and Joseph O'Rourke. The open problems project. http://maven.smith.edu/ orourke/TOPP/. 163

[dPP88] Hubert de Fraysseix, Janos Pach, and Richard Pollack. Small Sets Supporting Fáry Embeddings of Planar Graphs. In Janos Simon, editor, *Symposium on Theory of Computing (STOC '88)*, pages 426–433, 1988. 17

[dPP90] Hubert de Fraysseix, Janos Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. 17, 163

[dR82] Hubert de Fraysseix and Pierre Rosenstiehl. A depth-first-search characterization of planarity. *Annals of Discrete Mathematics*, 13:75–80, 1982. 12

[DT90] Giuseppe Di Battista and Roberto Tamassia. On-line graph algorithms with SPQR-trees. In Mike Paterson, editor, *International Colloquium on Automata, Languages and Programming (ICALP '90)*, LNCS, pages 598–611, 1990. 20

[DT96a] Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996. 20, 124

[DT96b] Giuseppe Di Battista and Roberto Tamassia. On-line planarity testing. *SIAM Journal on Computing*, 25(5):956–997, 1996. 20, 124

[Ede87] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1987. 7

[EFK01] Markus Eiglsperger, Sandor P. Fekete, and Gunnar W. Klau. Orthogonal graph drawing. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs*, volume 2025 of *Lecture Notes in Computer Science*, pages 121–171. Springer Berlin Heidelberg, 2001. 127

[EKP03] Cesim Erten, Stephen G. Kobourov, and Chandan Pitta. Intersection-free morphing of planar graphs. In Giuseppe Liotta, editor, *11th Symposium on Graph Drawing (GD'03)*, pages 320–331, 2003. 29

[ET76] Shimon Even and Robert E. Tarjan. Computing an *st*-numbering. *Theoretical Computer Science*, 2:339–344, 1976. 12

[Eve79] Shimon Even. *Graph Algorithms*. W. H. Freeman & Co, New York, USA, 1979. 7

[Fár48] István Fáry. On straight line representation of planar graphs. *Acta Universitaria Szegediensis, Sectio Scientiarum Mathematicarum*, 11:229–233, 1948. 16, 40, 46, 51, 81, 88, 91

[FCE95a] Qing-Wen Feng, Robert F. Cohen, and Peter Eades. How to draw a planar clustered graph. In Ding-Zhu Du and Ming Li, editors, *Computing and Combinatorics Conference (COCOON '95)*, volume 959 of *LNCS*, pages 21–30, 1995. 165

[FCE95b] Qing-Wen Feng, Robert F. Cohen, and Peter Eades. Planarity for clustered graphs. In Paul G. Spirakis, editor, *European Symposium on Algorithms (ESA '95)*, volume 979 of *LNCS*, pages 213–226, 1995. 165, 166, 169

[FE02] Carsten Friedrich and Peter Eades. Graph drawing in motion. *Journal of Graph Algorithms and Applications*, 6(3):353–370, 2002. 29

[FG99] Michael S. Floater and Craig Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics*, 101(1–2):117 – 129, 1999. 29

[FHK98] Ulrich Fößmeier, Carsten Heß, and Michael Kaufmann. On improving orthogonal drawings: The 4M-algorithm. In Sue H. Whitesides, editor, *Graph Drawing*, volume 1547 of *LNCS*, pages 125–137, 1998. 134

[FK95] Ulrich Fößmeier and Michael Kaufmann. Drawing high degree graphs with low bend numbers. In Roberto Tamassia and Ioannis G. Tollis, editors, *Graph Drawing*, volume 1027 of *LNCS*, pages 254–266, 1995. 127

[FKR05] Michael S. Floater, Géza Kós, and Martin Reimers. Mean value coordinates in 3D. *Computer Aided Geometric Design - Special issue: Geometric modelling and differential geometry*, 22(7):623–631, October 2005. 29

[Flo03] Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, March 2003. 29

[For05] Michael Forster. *Crossings in clustered level graphs*. PhD thesis, University of Passau, 2005. 169

[FP08] Fabrizio Frati and Maurizio Patrignani. A note on minimum area straight-line drawings of planar graphs. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Graph Drawing*, volume 4875 of *Lecture Notes in Computer Science*, pages 339–344. Springer Berlin Heidelberg, 2008. 17

[GH94] Leonidas Guibas and John Hershberger. Morphing simple polygons. In Kurt Mehlhorn, editor, *Proceedings of the Tenth Annual Symposium on Computational Geometry*, SOCG '94, pages 267–276, New York, NY, USA, 1994. ACM. 28

[GJL$^+$02] Carsten Gutwenger, Michael Jünger, Sebastian Leipert, Petra Mutzel, Merijam Percan, and René Weiskircher. Advances in c-planarity testing of clustered graphs. In Stephen G. Kobourov and Michael T. Goodrich, editors, *Graph Drawing*, volume 2528 of *LNCS*, pages 220–235. Springer, 2002. 165

[GLS05] Michael T. Goodrich, George S. Lueker, and Jonathan Z. Sun. C-planarity of extrovert clustered graphs. In Patrick Healy and Nikola S. Nikolov, editors, *Graph Drawing (GD '05)*, volume 3843 of *LNCS*, pages 211–222, 2005. 165

[GM01] Carsten Gutwenger and Petra Mutzel. A linear time implementation of SPQR-trees. In Joe Marks, editor, *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer Berlin Heidelberg, 2001. 22, 124

[GMPP91] Peter Gritzmann, Bojan Mohar, Janos Pach, and Richard Pollack. Embedding a planar triangulation with vertices at specified positions. *The American Mathematical Monthly*, 98(2):165–166, 1991. 163

[GS81] Branko Grünbaum and Geoffrey C. Shephard. The geometry of planar graphs. In Harold N. V. Temperley, editor, *Combinatorics, Proceedings of the eighth British combinatorial conference*, volume 52 of *London Mathematical Society Lecture Note Series*, pages 124–150. Cambridge University Press, 1981. 28

[GS01] Craig Gotsman and Vitaly Surazhsky. Guaranteed intersection-free polygon morphing. *Computers and Graphics*, 25:67–75, 2001. 29

[GT95] Ashim Garg and Roberto Tamassia. Upward planarity testing. *Order*, 12(2):109–133, 1995. 100

[GT01] Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM Journal on Computing*, 31(2):601–625, 2001. 127, 128

[GT09] Michael T. Goodrich and Roberto Tamassia. *Algorithm Design: Foundations, Analysis and Internet Examples*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2009. 7

[Har69] Frank Harary. *Graph theory*. Addison-Wesley, 1969. 7

[HEH09] Weidong Huang, Peter Eades, and Seok-Hee Hong. A graph reading behavior: Geodesic-path tendency. In Qinping Zhao and Hongbin Zha, editors, *PacificVis*, pages 137–144, 2009. 3, 100

[Ho73a] Chung-Wu Ho. On certain homotopy properties of some spaces of linear and piecewise linear homeomorphisms. I. *Transactions of the American Mathematical Society*, 181:213–233, 1973. 28

[Ho73b] Chung-Wu Ho. On certain homotopy properties of some spaces of linear and piecewise linear homeomorphisms. II. *Transactions of the American Mathematical Society*, 181:235–243, 1973. 28

[Ho74] Chung Wu Ho. Deforming p. l. homeomorphisms on a convex polygonal 2-disk. *Pacific Journal of Mathematics*, 55(2):427–439, 1974. 28

[Ho75] Chung-Wu Ho. Deforming p.l. homeomorphisms on a convex 2-disk. *Bulletin of the American Mathematical Society*, 81(4):726–728, Jul 1975. 28

[HP66] Frank Harary and Geert Prins. The block-cutpoint-tree of a graph. *Publicationes Mathematicae Debrecen*, 13:103–107, 1966. 19

[HR13] Md.Iqbal Hossain and Md.Saidur Rahman. Straight-line monotone grid drawings of series-parallel graphs. In Ding-Zhu Du and Guochuan Zhang, editors, *19th International Computing and Combinatorics Conference (COCOON ’13)*, volume 7936 of *Lecture Notes in Computer Science*, pages 672–679. Springer Berlin Heidelberg, 2013. 100

[HS95] John Hershberger and Subhash Suri. Morphing binary trees. In Kenneth L. Clarkson, editor, *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’95, pages 396–404, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics. 28

[HT74] John Hopcroft and Robert E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, October 1974. 12

[HT08] Bernhard Haeupler and Robert E. Tarjan. Planarity algorithms via PQ-trees (extended abstract). *Electronic Notes in Discrete Mathematics*, 31:143–149, 2008. 12

[IMH05] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics*, 24(3):1134–1141, July 2005. 29

[JJKL08] Vít Jelínek, Eva Jelínková, Jan Kratochvíl, and Bernard Lidický. Clustered planarity: Embedded clustered graphs with two-component clusters. In Maurizio Patrignani and Ioannis Tollis, editors, *Graph Drawing*, volume 5417 of *LNCS*, pages 121–132. Springer, 2008. 165

[JKK+07] Eva Jelínková, Jan Kára, Jan Kratochvíl, Martin Pergel, Ondrej Suchý, and Tomás Vyskocil. Clustered planarity: Small clusters in eulerian graphs. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Graph Drawing*, volume 4875 of *Lecture Notes in Computer Science*, pages 303–314. Springer, 2007. 165

[JMD+07] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics*, 26(3), July 2007. 29

[JSTV08] Vít Jelínek, Ondrej Suchý, Marek Tesar, and Tomás Vyskocil. Clustered planarity: Clusters with few outgoing edges. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Graph Drawing*, volume 5417 of *Lecture Notes in Computer Science*, pages 102–113. Springer, 2008. 165

[KCP92] James R. Kent, Wayne E. Carlson, and Richard E. Parent. Shape transformation for polyhedral objects. In James J. Thomas, editor, *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 47–54, New York, NY, USA, 1992. ACM. 28

[Kir88] David G. Kirkpatrick. Establishing order in planar subdivisions. *Discrete & Computational Geometry*, 3:267–280, 1988. 12

[KL08] Stephen G. Kobourov and Mattew Landis. Morphing planar graphs in spherical space. *Journal of Graph Algorithms and Applications*, 12(1):113–127, 2008. 29

[Kur30] Kazimierz Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930. 8, 10

[Kur04] Maciej Kurowski. A 1.235 lower bound on the number of points needed to draw all n-vertex planar graphs. *Information Processing Letters*, 92(2):95–98, 2004. 163

[KW01] Michael Kaufmann and Dorothea Wagner, editors. *Drawing Graphs: methods and models*. Lecture Notes in Computer Science. Springer, Berlin, New York, 2001. 7

[Lei80] Charles E. Leiserson. Area-efficient graph layouts (for VLSI). In *21st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 270–281. IEEE Computer Society, 1980. 127

[LM10] Tom Leighton and Ankur Moitra. Some results on greedy embeddings in metric spaces. *Discrete & Computational Geometry*, 44(3):686–705, 2010. 100

[LMS98] Yanpei Liu, Aurora Morgana, and Bruno Simeone. A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid. *Discrete Applied Mathematics*, 81(13):69 – 91, 1998. 127

[LP11] Anna Lubiw and Mark Petrick. Morphing planar graph drawings with bent edges. *Journal of Graph Algorithms and Applications*, 15(2):205–227, 2011. 28

[LPS06] Anna Lubiw, Mark Petrick, and Michael J. Spriggs. Morphing orthogonal planar graph drawings. In Cliff Stein, editor, *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 222–230, New York, NY, USA, 2006. ACM. 28

[MBLD02] Mark Meyer, Alan Barr, Haeyoung Lee, and Mathieu Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphic Tools*, 7(1):13–22, November 2002. 29

[Mei75] Gary H. Meisters. Polygons have ears. *The American Mathematical Monthly*, 82(6):648–651, 1975. 42, 43

[NMWB08] Martin Nöllenburg, Damian Merrick, Alexander Wolff, and Marc Benkert. Morphing polylines: A step towards continuous generalization. *Computers, Environment and Urban Systems*, 32(4):248 – 260, 2008. Geographical Information Science Research, United Kingdom. 28

[NR04] Takao Nishizeki and Md. Saidur Rahman. *Planar Graph Drawing*, volume 12 of *Lecture Notes Series on Computing*. World Scientific, Singapore, 2004. 7

[NW11] Martin Nöllenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):626–641, May 2011. 128

[OPG] Open problem garden. http://garden.irmacs.sfu.ca. 163

[PCA02] Helen C. Purchase, David A. Carrington, and Jo-Anne Allder. Empirical evaluation of aesthetics-based graph layout. *Empirical Software Engineering*, 7(3):233–255, 2002. 15

[PCJ97] Helen C. Purchase, Robert F. Cohen, and Murray I. James. An experimental study of the basis for graph drawing algorithms. *ACM Journal of Experimental Algorithmics*, 2:4, 1997. 9, 15

[PR05] Christos H. Papadimitriou and David Ratajczak. On a conjecture related to geometric routing. *Theoretical Computer Science*, 344(1):3–14, 2005. 100

[PS85] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction*. Springer, 1985. 7

[PT97] Achilleas Papakostas and Ioannis G. Tollis. A pairing technique for area-efficient orthogonal drawings (extended abstract). In Stephen North, editor, *Graph Drawing*, volume 1190 of *Lecture Notes in Computer Science*, pages 355–370. Springer Berlin Heidelberg, 1997. 127

[Pur00] Helen C. Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with Computers*, 13(2):147–162, 2000. 9, 15

[Rob81] David F. Robinson. The description of rectanguloid curves. In Harold N. V. Temperley, editor, *Combinatorics, Proceedings of the eighth British combinatorial conference*. Cambridge [Cambridgeshire] ; New York : Cambridge University Press, 1981. Lecture at The Eighth British Combinatorial Conference. 28

[Ros10] Vincenzo Roselli. Animazione di grafi: Morphing di strutture planari. Master's thesis, Roma Tre University, Rome, Italy, 2010. 29, 71

[Sch90] Walter Schnyder. Embedding planar graphs on the grid. In David S. Johnson, editor, *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 138–148, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics. 17, 163

[SG92] Thomas W. Sederberg and Eugene Greenwood. A physically based approach to 2-D shape blending. In James J. Thomas, editor, *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 25–34, New York, NY, USA, 1992. ACM. 28

[SG01] Vitaly Surazhsky and Craig Gotsman. Controllable morphing of compatible planar triangulations. *ACM Transactions on Graphics*, 20(4):203–231, October 2001. 29

[SG03] Vitaly Surazhsky and Craig Gotsman. Intrinsic morphing of compatible triangulations. *International Journal of Shape Modeling*, 09(02):191–201, 2003. 29

[Smi17] Herman L. Smith. On continuous representations of a square upon itself. *Annals of Mathematics*, 19(2):137–141, 1917. 27

[Spr07] Michael J. Spriggs. *Morphing parallel graph drawings*. PhD thesis, University of Waterloo, Waterloo, Candada, 2007. 28

[Ste58] Moritz A. Stern. Über eine zahlentheoretische funktion. *Journal für die reine und angewandte Mathematik*, 55:193–220, 1858. 101

[Ste16] Ernst Steinitz. Polyeder und raumeinteilungen. *Enzyklopädie der mathematischen Wissenschaften mit Einschluss ihrer Anwendungen*, III.AB(12):1–139, 1916. 27

[Ste51] Sherman K. Stein. Convex maps. *Proceedings of the American Mathematical Society*, 2:464–466, 1951. 16

[Tam87] Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal of Computing*, 16(3):421–444, 1987. 127, 129, 133, 147

[Tho83] Carsten Thomassen. Deformations of plane graphs. *Journal of Combinatorial Theory, Series B*, 34(3):244–257, 1983. 28

[Tie14] Heinrich Tietze. Über stetige abbildungen einer quadratfläche auf sich selbst. *Rendiconti del Circolo Matematico di Palermo*, 38(1):247–304, 1914. 27

[TT89] Roberto Tamassia and Ioannis G. Tollis. Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234, 1989. 127

[Tut63] William T. Tutte. How to draw a graph. *London Mathematical Society*, 13(3):743–768, 1963. 29

[Val81] Leslie G. Valiant. Universality considerations in VLSI circuits. *IEEE Transaction on Computers*, 30(2):135–140, 1981. 17, 127

[Veb17] Oswald Veblen. On the deformation of an n-cell. *Proceedings of the National Academy of Sciences of the United States of America*, 3(11):654–656, 1917. 27

[Wag36] Klaus Wagner. Bemerkungen zum vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 2:26–32, 1936. 16

[Wag37] Klaus Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114(1):570–590, 1937. 11

[WPCM02] Colin Ware, Helen C. Purchase, Linda Colpoys, and Matthew McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002. 15

[Yan89] Mihalis Yannakakis. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38(1):36 – 67, 1989. 164

[ZH03] Huaming Zhang and Xin He. Compact visibility representation and straight-line grid embedding of plane graphs. In Frank Dehne, Jörg-Rüdiger Sack, and Michiel Smid, editors, *Algorithms and Data Structures (WADS '03)*, volume 2748 of *LNCS*, pages 493–504, 2003. 17