Algorithms for Incremental Planar Graph Drawing and Two-page Book Embeddings

Inaugural-Dissertation
zur
Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität zu Köln

vorgelegt von
Martin Gronemann
aus Unna

Köln 2015

Berichterstatter (Gutachter):	Prof. Dr. Michael Jünger Prof. Dr. Markus Chimani Prof. Dr. Bettina Speckmann
Tag der mündlichen Prüfung:	23. Juni 2015

Zusammenfassung

Diese Arbeit beschäftigt sich mit zwei Problemen bei denen es um Knotenordnungen in planaren Graphen geht. Hierbei werden als erstes Ordnungen betrachtet, die als Grundlage für inkrementelle Zeichenalgorithmen dienen. Solche Algorithmen erweitern in der Regel eine vorhandene Zeichnung durch schrittweises Hinzufügen von Knoten in der durch die Ordnung gegebene Reihenfolge. Zu diesem Zweck kommen im Gebiet des Graphenzeichnens verschiedene Ordnungstypen zum Einsatz. Einigen dieser Ordnungen fehlen allerdings gewünschte oder sogar für einige Algorithmen notwendige Eigenschaften. Diese Eigenschaften werden genauer untersucht und dabei ein neuer Typ von Ordnung entwickelt, die sogenannte bitonische *st*-Ordnung, eine Ordnung, welche Eigenschaften kanonischer Ordnungen mit der Flexibilität herkömmlicher *st*-Ordnungen kombiniert. Die zusätzliche Eigenschaft bitonisch zu sein ermöglicht es, eine *st*-Ordnung wie eine kanonische Ordnung zu verwenden.

Es wird gezeigt, dass für jeden zwei-zusammenhängenden planaren Graphen eine bitonische *st*-Ordnung in linearer Zeit berechnet werden kann. Im Gegensatz zu kanonischen Ordnungen, können *st*-Ordnungen naturgemäß auch für gerichtete Graphen verwendet werden. Diese Fähigkeit ist für das Zeichnen von aufwärtsplanaren Graphen von besonderem Interesse, da eine bitonische *st*-Ordnung unter Umständen es erlauben würde, vorhandene ungerichtete Zeichenverfahren für den gerichteten Fall anzupassen. Basierend auf dieser Beobachtung wird eine Teilmenge der planaren *st*-Graphen charakterisiert, für die eine solche Ordnung existiert. Zusätzlich wird ein Linearzeit-Algorithmus vorgestellt, der diese Graphen erkennt und eine bitonische *st*-Ordnung berechnet. Für die übrigen planaren *st*-Graphen wird gezeigt, dass mittels Unterteilung spezifischer Kanten eine Instanz augmentiert werden kann, so dass für diese eine bitonische *st*-Ordnung existiert. Dabei bestimmt ein Linearzeit-Algorithmus die kleinste Kantenmenge für den Unterteilungsschritt. Weiterhin wird gezeigt, dass für einen planaren *st*-Graphen $G = (V, E)$ diese Menge nicht mehr als $|V| - 3$ Kanten umfasst und jede Kante höchstens einmal unterteilt werden muss. Dieses Resultat lässt sich sofort übertragen auf die Anzahl der benötigten Knicke in aufwärtsplanaren Zeichnungen. Es wird ein Algorithmus angegeben, der eine aufwärtsplanare Zeichnung mit quadratischer Fläche, höchstens $|V|-3$ Knicken insgesamt und maximal einem Knick pro Kante in linearer Zeit erstellt.

Der zweite Teil der Arbeit widmet sich der Einbettung von planaren Graphen

mit Maximalgrad drei und vier in Büchern mit zwei Seiten. Neben einem vereinfachten inkrementellen Linearzeit-Algorithmus für drei-zusammenhängende 3-planare Graphen, wird auch ein Linearzeit-Algorithmus für den 4-planaren Fall vorgestellt.

Abstract

Subject of this work are two problems related to ordering the vertices of planar graphs. The first one is concerned with the properties of vertex-orderings that serve as a basis for incremental drawing algorithms. Such a drawing algorithm usually extends a drawing by adding the vertices step-by-step as provided by the ordering. In the field of graph drawing several orderings are in use for this purpose. Some of them, however, lack certain properties that are desirable or required for classic incremental drawing methods. We narrow down these properties, and introduce the bitonic *st*-ordering, an ordering which combines the features only available when using canonical orderings with the flexibility of *st*-orderings. The additional property of being bitonic enables an *st*-ordering to be used in algorithms that usually require a canonical ordering.

With this in mind, we describe a linear-time algorithm that computes such an ordering for every biconnected planar graph. Unlike canonical orderings, *st*-orderings extend to directed graphs, in particular planar *st*-graphs. Being able to compute bitonic *st*-orderings for planar *st*-graphs is of particular interest for upward planar drawing algorithms, since traditional incremental algorithms for undirected planar graphs might be adapted to directed graphs. Based on this observation, we give a full characterization of the class of planar *st*-graphs that admit such an ordering. This includes a linear-time algorithm for recognition and ordering. Furthermore, we show that by splitting specific edges of an instance that is not part of this class, one is able to transform it into one for which then such an ordering exists. To do so, we describe a linear-time algorithm for finding the smallest set of edges to split. We show that for a planar *st*-graph $G = (V, E)$, $|V| - 3$ edge splits are sufficient and every edge is split at most once. This immediately translates to the number of bends required for upward planar poly-line drawings. More specifically, we show that every planar *st*-graph admits an upward planar poly-line drawing in quadratic area with at most $|V| - 3$ bends in total and at most one bend per edge. Moreover, the drawing can be obtained in linear time.

The second part is concerned with embedding planar graphs with maximum degree three and four into books. Besides providing a simplified incremental linear-time algorithm for embedding triconnected 3-planar graphs into a book of two pages, we describe a linear-time algorithm to compute a subhamiltonian cycle in a triconnected 4-planar graph.

Contents

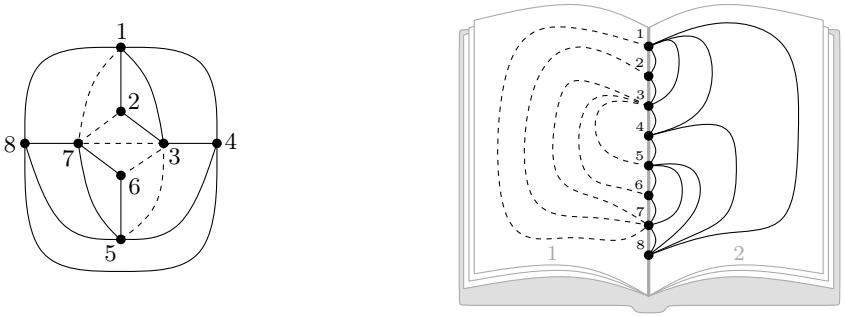
1 Introduction

Incremental drawing algorithms are a popular concept in the field of graph drawing. Their history goes back to the early beginnings of graph drawing, but even today, incremental approaches remain an important tool. Especially when dealing with problems that are concerned with planar graphs, various methods have been developed to solve these. Most algorithms follow a common principle that is similar to mathematical induction.

The first ingredient for an incremental drawing algorithm is an ordering of entities. These entities are usually the vertices, but also other orderings for sets of vertices, edges or faces exist. Once an ordering of the entities is determined, a suitable invariant is described that a partially constructed drawing must comply with. One starts by choosing a base case that is used to create an initial drawing which complies with the invariant. Afterwards, the entities are added step-by-step as provided by the ordering. In each step, one has to ensure that the drawing can be extended while at the same time the invariant is satisfied. Sometimes it is necessary to distinguish the placement of the last entity.

Algorithms that follow this abstract pattern have a long tradition, not only in the field of graph drawing. The type of ordering used depends heavily on the problem that has to be solved, and the ordering itself might require a certain type of graph. A well-known example are canonical orderings of which there exist multiple variants, each demanding different structural properties from the input graph. These serve as a basis for a variety of incremental drawing algorithms. In order to compensate for the restrictions imposed by the type of ordering used, one may decompose the graph into components that fulfill the requirements, compute drawings for these components and compose their drawings afterwards.

In this thesis we investigate the properties of orderings that are suitable for incremental drawing algorithms in more detail, and propose a new type of ordering, the bitonic *st*-ordering. The backstory that eventually led to this idea is as follows. In [5] and [6] we extend existing incremental drawing algorithms that require the input to be triconnected to ones that require only a biconnected graph using a standard decomposition technique that is referred to as SPQR-tree approach. It follows the described principle of decomposing the graph and merging the drawings afterwards. In [6] the input graph is further restricted to have maximum degree of three, the so-called 3-planar graphs, whereas in [5]



(a) (b)

Figure 1.1. (a) A planar graph embedded in the plane and (b) a two-page book embedding of the same graph. Edges drawn dashed are assigned to the first page, whereas solid ones are drawn on the second page.

a different problem is solved that is concerned with 4-planar graphs. In both cases, the degree restriction is of importance. While SPQR-trees on 3-planar graphs have a simple well-known structure which can easily be exploited, for 4-planar graphs this is not the case. As a result, one has to distinguish many subcases, thereby, turning a rather simple triconnected algorithm into a complex biconnected one.

Motivated by trying to avoid the SPQR-tree approach, we study the properties of canonical orderings that are required by most incremental drawing algorithms. The result is a new ordering that preserves those properties, but is not as restrictive in terms connectivity. Moreover, we show that for a restricted class of directed graphs, one may obtain such an ordering as well. This is of particular interest from an algorithmic point of view, because canonical orderings do not extend to directed graphs at all.

The second topic of this thesis that is not directly related to incremental drawing algorithms, is the problem of embedding graphs into books. A book embedding is a special form of embedding in which the vertices are placed along a line, the spine of a book, and each edge is drawn entirely on one page of a book such that it crosses no other edge on the same page. An example for a two-page book embedding of a graph is displayed in Figure 1.1. When considering Figure 1.1b, it is not hard to see that a two page book embedding is a drawing in the plane in which no two edges cross, while at the same time the vertices are aligned. However, not every planar graph admits such a book embedding. Therefore, it has always been of interest which subsets of the planar graphs fit into two pages. With this in mind we investigate bounded degree graphs, and show that the 4-planar graphs admit such an embedding.

Outline

We begin in Chapter 2 with the notation and definitions that serve as a basis for all topics covered in this thesis. Following this, we give a detailed introduction into drawing conventions for directed planar graphs. Besides introducing a few existing drawing styles, we derive some insights into the classes of graphs that admit those. Some results in this part are motivated by questions that arose during the *5th International Conference on Information, Intelligence, Systems and Applications (IISA 2014)*.

Chapter 3, the most comprehensive one, introduces the bitonic *st*-ordering and can be roughly divided into two parts. The first part contains related work, definitions and is concerned with bitonic *st*-orderings of undirected biconnected planar graphs. It is based on a conference paper [50] presented at the *International Symposium on Graph Drawing (GD 2014)*, whereas the second part takes this idea one step further, and applies it to directed graphs, in particular planar *st*-graphs. The contribution of this chapter is, besides the introduction of the bitonic *st*-ordering, linear-time algorithms, for both the undirected and directed case including a recognition algorithm for the latter case. Applying this idea to planar *st*-graphs, yields a new upper bound on number of bends in upward planar poly-line drawings.

We study in Chapter 4 the problem of embedding planar graphs of bounded degree into books with two pages. After reviewing related work, we turn our attention for the time being to triconnected planar graphs with maximum degree three, the so called 3-planar graphs, and describe a canonical ordering-based incremental algorithm to solve the problem. Although no new results are derived, the algorithm is interesting from a practical point of view. Afterwards, we investigate the structure of separating triangles in the 4-planar case and show that due to their special properties, every triconnected 4-planar graph can be embedded into a book of two pages. The proof is constructive and yields a linear-time algorithm. The 4-planar case is based on the first part of a paper [7, 8] that has been presented at the *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)* and is joint work with Michael Bekos and Chrysanthi Raftopoulou. The second contribution of the paper, that is the general 4-planar case, is not part of this thesis, but a brief overview of the idea is given at the end of the chapter. Work on this problem began in 2013 during *Dagstuhl Seminar 13151: Drawing Graphs and Maps with Curves*. A short conclusion of the main results is given in Chapter 5.

2 Preliminaries

We start with fundamental definitions and notation that are used throughout this work, details are introduced in the corresponding chapters. However, we assume that the reader is familiar with the basics of computer science, especially algorithms. The purpose of this chapter is two-fold. The first part is concerned with the very basic elements of graph drawing and follows partially [42, 59]. The second part discusses various drawing styles for directed graphs in more detail and from a geometric point of view.

2.1 Technical foundations

A *graph* $G = (V, E)$ consists of a finite set V called the *vertices* or *nodes* of G, and a finite set of *edges* $E \subseteq V \times V$ being a multi-set of pairs $(u, v) \in E$ with $u, v \in V$. We sometimes refer to the vertices V of G by $V(G)$, and to the set of edges E by $E(G)$. Given an edge $e = (u, v) \in E$, u and v are referred to as the *endpoints* of e. Furthermore, we say that u and v are *incident* to the edge e, and that u is a *neighbor* of v and vice versa. If u and v are neighbors, then we may also say that u and v are *adjacent*.

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs with $V' \subseteq V$ and $E' \subseteq E$, then G' is a *subgraph* of G which we denote by $G' \subseteq G$. In case G' and G share the same vertex set, that is, $V = V'$ holds, we may emphasize this by calling G' a *spanning* subgraph of G. Let $S \subseteq V$ be a subset of the vertices of G. Then S *induces* the subgraph $(S, E \cap (S \times S))$, that is the subgraph of G consisting of S and all the edges having their endpoints in S. Furthermore, we may write $G - S$ for the graph $G' = (V', E')$ with $V' = V - S$ and $E' = E \cap (V' \times V')$, that is, G' is obtained by removing all vertices in S from G including their incident edges. Moreover, for two graphs $G = (V, E)$ and $G' = (V', E')$, we define $G \cup G' = (V \cup V', E \cup E')$ and $G \cap G' = (V \cap V', E \cap E')$.

In an *undirected* graph, every edge $(u, v) \in E$ is considered to be an unordered pair, whereas in a *directed* graph or for short *digraph*, (u, v) is an ordered pair. In the literature an undirected edge is sometimes denoted by $\{u, v\}$ to emphasize that it is an undirected edge. Since we do not mix directed and undirected edges, and usually from the context it is clear whether the corresponding graph is directed or undirected, we use here in both cases the notation (u, v) to refer

to an edge. Most of the definitions that follow work for both types, and if that is not the case or the context is not clear, we explicitly distinguish between them. Some definitions, however, require a directed graph to be considered undirected. Hence, we refer to the graph that is obtained by ignoring the direction of the edges of a directed graph, as the corresponding *underlying undirected* graph.

If $G = (V, E)$ is directed, then we may refer to $e = (u, v) \in E$ as an *outgoing* edge of u and an *incoming* edge of v. Furthermore, v is a *successor* of u and u is a *predecessor* of v. The *degree* $\deg(v)$ of a vertex v is the number of incident edges of v. The number of outgoing edges of a vertex is referred to as the *out-degree*. In a symmetric manner, the number of incoming edges is called the *in-degree*. In particular, a vertex having no predecessors is a *source*, whereas a *sink* is a vertex without successors. An edge $e = (u, v)$ with $u = v$ is called a *self-loop*. Two edges $(u, v), (u', v')$ with $u = u'$ and $v = v'$ are said to be *parallel*. If G contains neither self-loops nor parallel edges, then G is *simple*. A simple undirected graph that contains every possible edge between every pair of vertices is called a *complete graph*. The complete graph on n vertices is denoted by K_n.

Paths & cycles For two vertices $s, t \in V$, a *path* from s to t is an alternating sequence of vertices and edges $v_0, e_1, v_1, \ldots, v_{k-1}, e_k, v_k$ with $v_0 = s$, $v_k = t$, and for every $1 \leq i \leq k$, $e_i = (v_{i-1}, v_i) \in E$ holds. We refer to the first vertex v_0 and last vertex v_k as the *endpoints* of the path, whereas the remaining ones $v_1, \ldots, v_{k-1}$ are called *inner* vertices. A set of paths that have no inner vertices in common, are said to be *independent*. The length of a path is the number of edges k it contains. A path is *simple*, if all vertices on the path are distinct. If a simple path has length $|V| - 1$, then it visits every vertex exactly once. Such a path is called a *Hamiltonian* path.

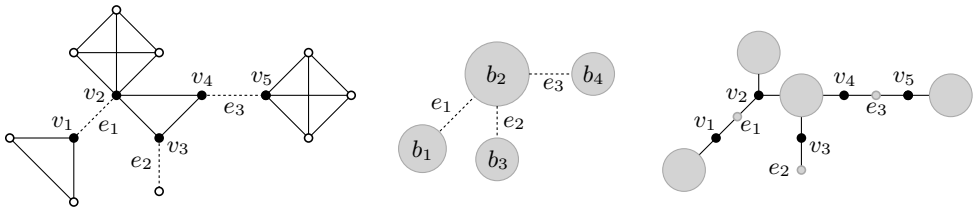
We denote a simple path from s to t in G by $s \rightsquigarrow t \in G$. If in the corresponding context G is clear without ambiguity, we may omit G, and just write $s \rightsquigarrow t$ to express that there exists a path from s to t in G. Conversely, if there exists no path from s to t, we denote this by $s \not\rightsquigarrow t$. If a path $s \rightsquigarrow t$ consists only of a single edge, we sometimes emphasize this by writing $s \to t$. Moreover, we may concatenate these expressions, for example, $s \rightsquigarrow u \to v \rightsquigarrow t$ is a path from s to t via u, v with u and v appearing consecutively, that is, the path contains the edge (u, v). If there exists a path from $u \rightsquigarrow v \in G$ and an edge $e = (v, u)$ such that the path does not contain e, then the path plus e is called a *cycle*. We often denote with $C = \{v_1, \ldots, v_k\}$ a cycle that contains the vertices $v_1, \ldots, v_k$ and assume that there exists for every $1 \leq i < k$ an edge $(v_i, v_{i+1}) \in E$ and $(v_k, v_1) \in E$. Similar to a simple path, the vertices on a *simple* cycle are all

distinct. The length of a simple cycle is the number of edges it contains, which is equal to the number of its vertices. Furthermore, we refer sometimes to a simple cycle of length k as *k-cycle*, and in case the simple cycle has length $|V|$, that is, it contains all vertices of G, we call it a *Hamiltonian* cycle. A graph that does not contain a cycle is *acyclic*, and in case of an undirected graph, then usually called a *forrest*. If a digraph is acyclic and contains exactly one source and one sink, we refer to it as *st-graph*.

Connectivity In this thesis, connectivity is not used together with digraphs, thus, the following terms are only being defined for undirected graphs. Let $G = (V, E)$ be a simple undirected graph. If for every pair $u, v \in V$ with $u \neq v$, $u \rightsquigarrow v \in G$ holds, then G is *connected*. Furthermore, G is *k-connected* for some integer $1 \leq k < |V|$, if for every subset of vertices $S \subset V$ with $|S| < k$, $G - S$ is connected. Instead of 2-connected or 3-connected, one usually uses the terms *biconnected* or *triconnected*, respectively. A fundamental theorem by Menger [64] states that a graph is k-connected, if and only if there exist k independent paths between every pair of distinct vertices. An undirected graph that is connected and acyclic is called a *tree*. In the case of a directed graph that contains a single source and whose underlying undirected graph is connected and acyclic, we call it a *rooted tree*. The single source is referred to as the *root* of the tree, whereas the sinks are called the *leaves*. The successors of a node v are called the *children* of v, and the predecessor the *parent* of v.

Let G be a connected graph. If G is not biconnected, then there exists a so-called *cut vertex* or *articulation point* $v \in V$ such that $G - \{v\}$ is not connected. If there exists an edge $e \in E$ such that its removal disconnects G, then we call this edge a *bridge*. See Figure 2.1a for an example of cut vertices and bridges. In case G does not contain any bridges, it is said to be *bridge-less*. If G is biconnected but not triconnected, then a pair $\{u, v\} \subset V$ for which $G - \{u, v\}$ is not connected anymore, is referred to as a *separation pair*. In case G is not 4-connected and there exists a 3-cycle $C = \{v_1, v_2, v_3\}$ in G, whose removal disconnects G, that is, $G - C$ is not connected, then C is called a *separating triangle*. Notice that in difference to a separation pair, the vertices of a separating triangle have to be adjacent.

The maximal bridge-less components of a connected graph $G = (V, E)$ are called the *bridge-blocks* of G, whereas the maximal biconnected subgraphs of G are the *biconnected components*. Notice that a bridge-block is not necessarily biconnected. Let $B_1, \ldots, B_m \subseteq G$ be the bridge-blocks of G, and $E' \subseteq E$ the bridges connecting them. The bridge blocks are vertex disjoint, that is, $V(B_i) \cap V(B_j) = \emptyset$ holds for every $1 \leq i \neq j \leq m$. It is not difficult to see



(a) (b) (c)

Figure 2.1. (a) A connected graph with five cut vertices $v_1, \ldots, v_5$ (black) and three bridges e_1, e_2, e_3 (dotted). (b) The corresponding bridge-block tree with four bridge-blocks (grey). (c) The BC-tree with five C-nodes (black) representing the cut vertices and seven B-nodes (grey), three of them being bridges.

that the bridges connecting the bridge-blocks induce a tree structure, since by definition a bridge cannot be part of a cycle. The *bridge-block tree* $\mathcal{T}$ is the tree that results from considering the bridge-blocks as nodes, say $V(\mathcal{T}) = \{b_1, \ldots, b_m\}$ and the bridges as edges connecting them, that is, if there exists a bridge $(u, v) \in E'$ with $u \in V(B_i) \wedge u \in V(B_j)$, then there exists an edge $(b_i, b_j) \in E(\mathcal{T})$. Figure 2.1b shows an example for a bridge-block tree. The construction of a bridge-block tree takes linear time [67].

In a similar manner, we define the *biconnected components tree (BC-tree)*. Let $B_1, \ldots, B_m$ be now the biconnected components of G. Unlike bridge-blocks, biconnected components are not vertex disjoint, but edge disjoint, that is, $E(B_i) \cap E(B_j) = \emptyset$ holds for every $1 \leq i \neq j \leq m$. However, two biconnected components may share at most one vertex which is then a cut vertex. From its definition, it follows that a cut vertex is part of at least two biconnected components. Let $v_1, \ldots, v_k \in V$ be the set of cut vertices in G. The nodes in a BC-tree $\mathcal{T}$ are of two types: B-nodes and C-nodes. The B-nodes $B = \{b_1, \ldots, b_m\}$ represent the biconnected components $B_1, \ldots, B_m$, whereas the C-nodes $C = \{c_1, \ldots, c_k\}$ represent the cut vertices $v_1, \ldots, v_k$. A B-node representing B_i is adjacent to a C-node representing a cut vertex v_j, if v_j is part of B_i, that is, $E(\mathcal{T}) = \{(b_i, c_j) \mid v_j \in V(B_i)\}$. The BC-tree of the graph in Figure 2.1a is shown in Figure 2.1c.

An *SPQR-tree* $\mathcal{T}$ reflects the decomposition of a biconnected graph $G = (V, E)$ with $|E| > 2$ into its *triconnected components* and their relationships [38, 52]. In fact, every triconnected component $G_\mu = (V_\mu, E_\mu)$ is represented by a tree node μ in $\mathcal{T}$ where G_μ itself is called the *skeleton* of μ. The interrelationship between two triconnected components is described by a pair of so-called *virtual edges*. Both virtual edges share the same endpoints that correspond to a *split pair* $\{s, t\}$. A split pair $\{s, t\}$ is either a pair of adjacent nodes in G or a separation

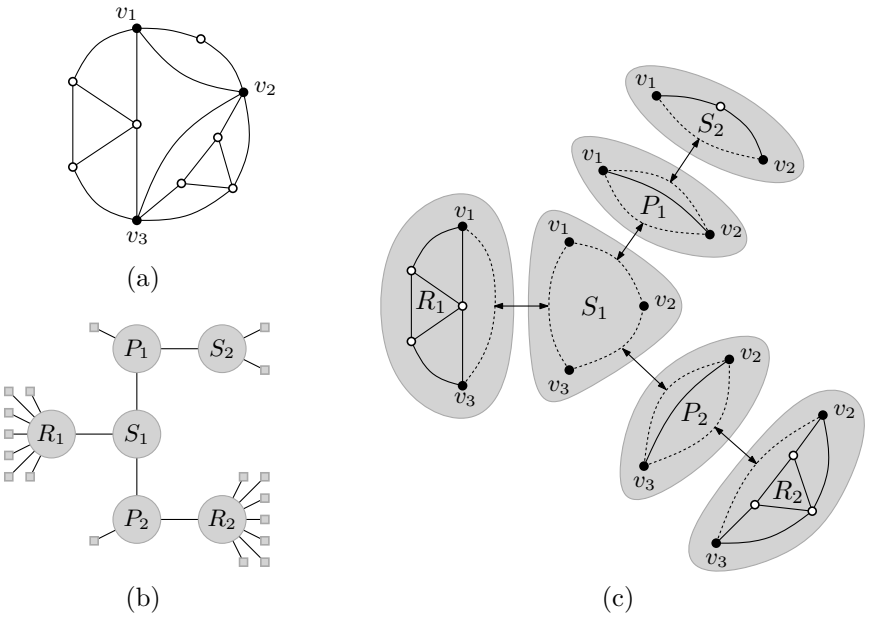


Figure 2.2. (a) A biconnected graph that is not triconnected with v_1, v_2, v_3 forming separation pairs. (b) The corresponding SPQR-tree consisting of two R-nodes R_1, R_2, two S-nodes S_1, S_2, two P-nodes P_1, P_2 and 18 Q-nodes drawn as small grey squares. (c) The same SPQR-tree, but with the skeletons of the nodes that are not Q-nodes. Virtual edges between them are drawn dotted, whereas the ones representing Q-nodes are drawn solid. The arrows indicate the relation between a pair of virtual edges.

pair. Every G_μ can be categorized to be one of four types based on its structure. A bundle of at least three parallel edges is referred to as *P-node*. In case G_μ is a simple cycle of length at least three, it classifies as an *S-node*, whereas if the skeleton is a simple triconnected graph, we call it an *R-node*. The leaves of $\mathcal{T}$ are formed by *Q-nodes* that are bundles of two edges, one being a virtual edge, while the other corresponds to an edge of G. An example for a small graph, its SPQR-tree and the skeletons is shown in Figure 2.2. Usually it is convenient to root $\mathcal{T}$, hence, inducing a hierarchy on the triconnected components. Except for the root, every skeleton G_μ contains then a virtual edge $(s,t) \in E_\mu$ that represents a link to μ's parent. We refer to (s,t) as the *reference edge* of μ and to its endpoints $\{s,t\}$ as the *poles* of μ. When considering a node μ in a rooted SPQR-tree $\mathcal{T}$, μ induces a subgraph of G called the *pertinent graph* of μ. Similar to the previous decompositions, an SPQR-tree can be constructed in linear time [52]. If the SPQR-tree of a biconnected graph contains no R-nodes, we may refer to it as *series-parallel*.

Drawings In a drawing $\Gamma \subset \mathbb{R}^2$ of a graph $G = (V, E)$, each vertex $v \in V$ is mapped to a distinct point $\Gamma(v)$ in the plane. Every edge $e = (u, v) \in E$ is mapped to an open jordan curve $\Gamma(e)$ starting at $\Gamma(u)$ and ending at $\Gamma(v)$ and these are the only two vertices that have a point in common with $\Gamma(e)$. In case G is directed, we also consider the curve of an edge (u, v) as being directed from $\Gamma(u)$ to $\Gamma(v)$. More precisely, for two points $p, q \in \Gamma(e)$, we denote with $p \prec q$ that p precedes q on $\Gamma(e)$. Notice that a path $u \rightsquigarrow v$ with $u \neq v$ induces an open jordan curve as well. Therefore, we may also use $p \prec q$ to express that there exists a path that induces an open jordan curve on which p precedes q. More specifically, there exists either an edge e such that $p \prec q$ holds or a path $u \to u' \rightsquigarrow v' \to v$ (possibly $u' = v'$) such that $p \in \Gamma((u, u'))$ and $q \in \Gamma((v', v))$.

If for every edge $e \in G$, $\Gamma(e)$ is a straight-line segment, instead of an arbitrary open jordan curve, then Γ is called a *straight-line* drawing. Similarly, if every edge is represented by a poly-line, we refer to Γ as a *poly-line* drawing. In that case every edge consists of a sequence of straight-line segments such that consecutive segments share an endpoint. Such an endpoint is called a *bend point*, or *bend* for short. If the coordinates of the vertices and possible bends are all integer, we refer to the drawing as *grid drawing*. The *width*, *height* and *area* of such a grid drawing correspond to the width, height and area of the smallest axis-aligned rectangle that contains it.

Planarity & embeddings Two edges *cross* in a drawing Γ, if their corresponding curves intersect. A *planar drawing* Γ of a graph $G = (V, E)$ is a drawing in which no two edges cross. A graph that admits a planar drawing is called a *planar graph*. Of particular interest in this work are planar graphs of bounded degree, that is, for some integer $k > 0$, $\deg(v) \leq k$ holds for every $v \in V$. We refer to these graphs as *k-planar*, that is, planar graphs with maximum degree at most k. A planar drawing of a graph G implies a circular ordering of the edges around every vertex, which we refer to as a *planar embedding* of G. Furthermore, a planar drawing subdivides the plane into topologically connected regions that are called the *faces*. If G is connected, every face is bounded by a not necessarily simple cycle of edges. We say a vertex or edge is *incident* to a face, if the vertex or edge is part of the bounding cycle, and two faces are *adjacent* if they share an edge. Notice that the region of exactly one face is unbounded, which is referred to as the *outer face* or *exterior face*, whereas the remaining faces are the *inner* or *interior faces*. In general, we assume that the outer face is given when referring to an embedding. A planar graph whose vertices are all incident to the outer face is said to be *outerplanar*. A planar graph $G = (V, E)$ is *maximal planar*, if one cannot add an edge without de-

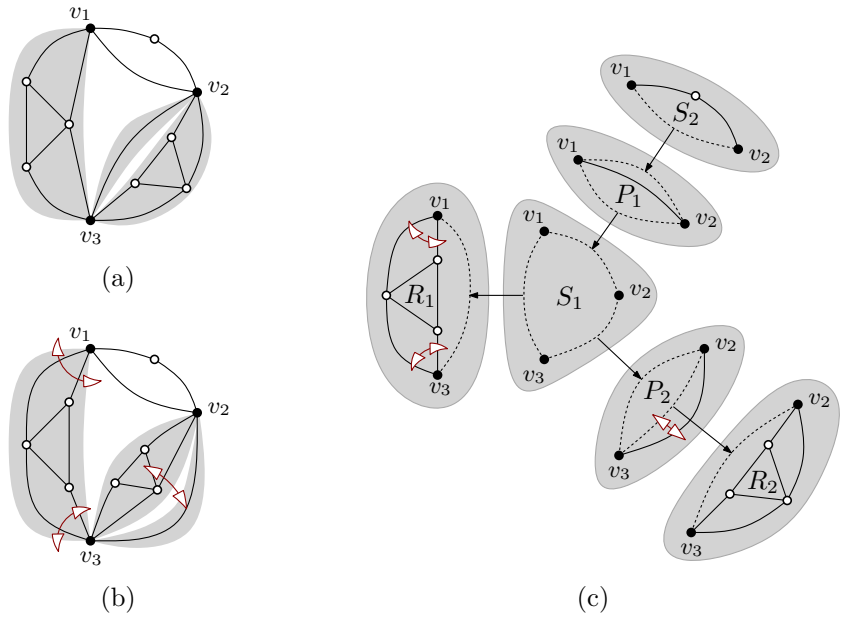


Figure 2.3. (a) Initial embedding of the planar graph from Figure 2.2a. (b) Embedding after flipping the induced subgraph of R_1 along the poles $\{v_1, v_3\}$ and swapping the one of R_2 with the Q-node in P_2. (c) The corresponding operations in the skeletons.

stroying planarity. Every face in a maximal planar graph is a 3-cycle and the graph is triconnected. Furthermore, for the number of edges and vertices in a maximal planar graph, it holds that $|E| = 3|V| - 6$ [42]. The *dual graph* of an embedded planar graph, is the (not necessarily simple) planar graph obtained from considering the faces as vertices and two vertices are adjacent in the dual graph, if the corresponding faces have an edge in common.

Now let $G = (V, E)$ be an st-graph. If $G' = (V, E \cup (s, t))$ is planar, then we say that G is a *planar st-graph* or has the property of being *st-planar*. Notice that in the literature, one may encounter a definition of st-planar in which it is assumed that the edge (s, t) is present. We explicitly do not require this property here and are satisfied with s and t being incident to a common face.

Testing whether an undirected graph is planar or not, can be accomplished in linear-time, and in case a planar embedding exists, it can be obtained within the same time frame [18, 24, 58, 63]. However, such an embedding is not unique. In particular, a planar graph that is not triconnected may have an exponential number of planar embeddings, whereas a triconnected graph has a unique embedding upon the choice of the outer face and reversing the circular ordering, that is, mirroring the embedding. The intuition behind this property is

the following: Consider a separation pair $\{u, v\}$ in a biconnected planar graph $G = (V, E)$. These may act like two hinges meaning that one may mirror one of the two subgraphs, while the other one maintains its orientation.

With this in mind, SPQR-trees are of particular interest, because they provide the information about the triconnected components of a graph. Every planar embedding induces a planar embedding of the skeletons and vice versa. Consider a tree node μ of $\mathcal{T}$ and its pertinent graph, say G'_μ, which is a subgraph of G. Now we may obtain a different embedding by *flipping* G'_μ along the poles of μ. While this works regardless of the node-type of μ, P-nodes offer even more possibilities. Recall that the skeleton of a P-node is a bundle of at least three virtual edges (one being the reference edge). Every ordering of the parallel virtual edges results in a different embedding. An example is given in Figure 2.3, in which the pertinent graph of R_1 is flipped along the poles, whereas in the skeleton of P_1 an edge-swap is performed. Now it is not hard to see that with these two operations one may generate an exponential number of different planar embeddings.

st-ordering Let $G = (V, E)$ be an undirected or directed graph with $s, t \in V$, $s \neq t$ and let $\pi : V \to \{1, \ldots, |V|\}$ be the rank of the vertices in an ordering $s = v_1, v_2, \ldots, v_n = t$, that is, $\pi(v_i) = i$ with $1 \leq i \leq n$. π is called an *st-ordering*, if for all vertices $v \in V$ with $1 < \pi(v) < n$, there exists $(u, v), (v, w) \in E$ with $\pi(u) < \pi(v) < \pi(w)$. It is not hard to see that a directed graph that admits such an *st*-ordering is an *st*-graph and vice versa. In order to obtain an *st*-ordering for an *st*-graph, it is sufficient to topologically sort the vertices, which takes linear time [31]. An undirected graph $G = (V, E)$ admits an *st*-ordering, if and only if, $G' = (V, E \cup (s, t))$ is biconnected. There exist several linear-time algorithms to compute an *st*-ordering in the undirected case, see e.g. [20, 46, 47]. Given an *st*-ordering π for G, we refer to the directed graph that is obtained by orienting every edge $(u, v) \in E$ such that $\pi(u) < \pi(v)$ holds, as the *st-orientation* of G.

2.2 Oriented planar drawings of directed graphs

When drawing directed graphs, one usually desires a drawing in which the edges flow in a common direction, for example from bottom to top. Of course this is not always possible. Consider a simple cycle; in its nature, there exists no common direction for the edges. The literature contains quite an extensive amount of work on this subject. Most of these drawing styles have in common that they require the drawing to be planar.

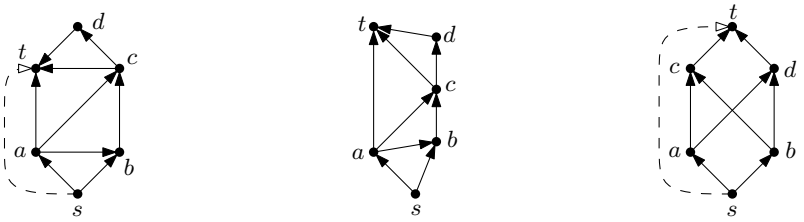


Figure 2.4. (a) A planar *st*-graph and (b) a corresponding upward planar straight-line layout. (c) A graph that does not admit an upward planar drawing and is not *st*-planar. After adding the dashed edge, the underlying undirected graph is not planar.

Let us formalize the idea of such oriented drawings in more detail. In the following, we introduce different conventions for drawing directed planar graphs. Instead of an algorithmic approach to this area, we focus on the relation of these drawings with the aim to sketch the landscape of directed graphs that admit such drawings. Besides existing styles, we introduce a new type of drawing whose sole purpose is to provide a high degree of freedom from a geometric point of view. These so-called not-downward drawings generalize in some sense the concept of drawing edges in a common direction. Although this generalization offers great flexibility, we show that the graphs that admit such drawings are the upward planar graphs that we will introduce now.

The so-called **upward planar** drawings are a type of drawing that has received much attention in the past. Given a planar drawing Γ of a directed graph $G = (V, E)$, we say Γ is upward planar, if for every edge $e \in E$, the corresponding curve $\Gamma(e)$ is strictly monotonically increasing in the vertical direction. More precisely, Γ is upward planar if for every $p, q \in \Gamma$, the following holds:

$$p \prec q \quad \Rightarrow \quad y(p) < y(q).$$

A directed graph G is called *upward planar*, if there exists a drawing of G that is upward planar. An example of a planar *st*-graph and an upward planar straight-line drawing is shown in Figure 2.4a and 2.4b, respectively. But not every planar digraph is upward planar. An example is depicted in Figure 2.4c. Deciding whether a planar acyclic digraph is upward planar is a difficult problem. Garg and Tamassia [49] show that the general case in which one may choose the embedding, is NP-complete. Several approaches to solve this problem have been developed, see e.g. [25, 26]. If the embedding is fixed or the graph contains only a single source, the problem is polynomial-time solvable [1, 12, 13, 35].

The following theorem by Di Battista and Tamassia [37] is fundamental when it comes to the characterization of upward planar graphs and drawings.

▶ **Theorem 2.1.** *[37] Given a planar digraph G, the following three statements are equivalent.*

- *G is upward planar.*
- *G admits an upward planar straight-line layout.*
- *G is a spanning subgraph of a planar st-graph.*[1]

Another type of straight-line drawing that is more restrictive than the upward planar one, is called **dominance** drawing. Here the edges are drawn upward and rightward. Moreover, for two vertices $u, v \in V$, we say that v *dominates* u, if and only if $x(u) \leq x(v)$ and $y(u) \leq y(v)$ holds. The idea of dominance drawings is that if there exists a path from u to v, then v dominates u in the drawing and vice versa. An example is shown in Figure 2.5a. In a more formal manner, we say that a planar straight-line drawing Γ is a dominance drawing, if for every pair $u, v \in V$ the following holds:

$$u \rightsquigarrow v \quad \Leftrightarrow \quad x(u) \leq x(v) \ \wedge \ y(u) \leq y(v).$$

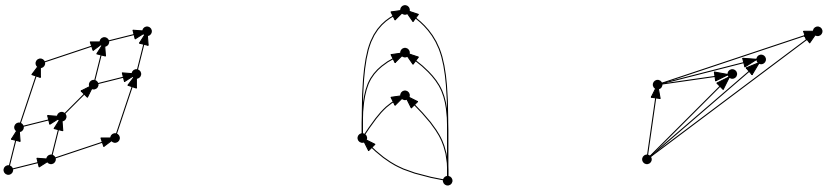
The condition that this holds both ways is quite strong compared to the requirements for an upward planar drawing. For example, the graph in Figure 2.5b does not admit a dominance drawing, but the depicted drawing is clearly upward planar.

Di Battista et al. [39] describe a linear-time algorithm to produce a dominance grid drawing within quadratic area. However, it requires a reduced planar *st*-graph as input. We introduce this subset of planar *st*-graphs in a more formal manner for reasons that will become evident later. In a planar *st*-graph $G = (V, E)$, an edge $(u, v) \in E$ is a *transitive edge*, if there exists a path $u \rightsquigarrow v$ in G that does not contain (u, v). If G does not contain transitive edges, then G is called a *reduced planar st-graph*. The algorithm in [39] does not employ any sophisticated techniques, but relies heavily on the fact that the input graph does not contain transitive edges.

A slightly relaxed version are the **weak dominance** drawings in which the existence of a path still implies the dominance relation, but not necessarily the other way around. More precisely, for every $u, v \in V$, the following holds:

$$u \rightsquigarrow v \quad \Rightarrow \quad x(u) \leq x(v) \ \wedge \ y(u) \leq y(v).$$

[1] Notice that the removal of vertices does not destroy upward planarity, therefore, a non-spanning subgraph of a planar *st*-graph is upward planar as well.



(a) (b) (c)

Figure 2.5. (a) Example for a dominance drawing. (b) An upward planar graph that does not admit a dominance drawing, but a weak dominance drawing (c).

Figure 2.5c shows a weak dominance drawing of the graph from Figure 2.5b, which does not admit a dominance drawing. By definition, it follows that every dominance drawing is a weak one. Assuming a weak dominance straight-line drawing Γ is given, rotating Γ counter-clockwise by an angle $0 < \alpha < \frac{\pi}{2}$ yields an upward planar straight-line drawing. This rotation is only necessary in case Γ contains horizontal edges which are not allowed in an upward drawing.

Although one may get the impression that the weak dominance property is slightly stricter than upward planarity, it turns out that this is not the case.

▶ **Lemma 2.2.** *A digraph G admits a planar weak dominance drawing if and only if G is upward planar.*

Proof. We only prove "⇐", the other direction has already been argued. Since $G = (V, E)$ is upward planar, it admits an upward planar straight-line drawing Γ. We show how to transform Γ into a planar weak-dominance straight-line drawing Γ'. The idea is to stretch Γ vertically such that for every edge $(u, v) \in E$, $|x(v) - x(u)| \leq y(v) - y(u)$ holds, that is, every edge $(u, v) \in E$ has an angle between $\frac{\pi}{4}$ and $\frac{3\pi}{4}$. Rotating the result by $\frac{\pi}{4}$ clockwise yields a weak dominance drawing. In order to preserve possible grid coordinates in Γ, an additional scaling of $\sqrt{2}$ is applied.

More specifically, let c be the vertical stretching factor for which we choose

$$c = \max_{(u,v)\in E} \left\lceil \frac{|x(v) - x(u)|}{y(v) - y(u)} \right\rceil .$$

We may assume that there exists at least one edge $(u, v) \in E$ with $x(u) \neq x(v)$, because otherwise, when all edges are vertical segments, Γ is already a planar weak-dominance straight-line drawing. Rounding up to the nearest integer is only necessary in case integer coordinates must be preserved. Putting it all

together, yields a new set of coordinates for Γ':

$$x'(v) = x(v) + c \cdot y(v) \quad \text{and} \quad y'(v) = c \cdot y(v) - x(v)$$

for every $v \in V$. In order to verify that the result is a weak dominance drawing, that is, $x'(v) - x'(u) \geq 0$ and $y'(v) - y'(u) \geq 0$ holds for every edge (u, v), we plug x' and y' into the above inequalities and solve for c. For the former, we get $c \geq \frac{x(u)-x(v)}{y(v)-y(u)}$, whereas for the latter, $c \geq \frac{x(v)-x(u)}{y(v)-y(u)}$ is obtained, which hold both due to our choice of c. Furthermore, the transformation is affine, therefore, planarity is not an issue. Notice that if $x(v)$ and $y(v)$ are integer, then the same holds for $x'(v)$ and $y'(v)$, because c is integer. In case Γ is a grid drawing, that is all coordinates are integer, then it is not difficult to see that c is bounded by the width of Γ, which enables us to preserve a possible polynomial area property of the drawing. □

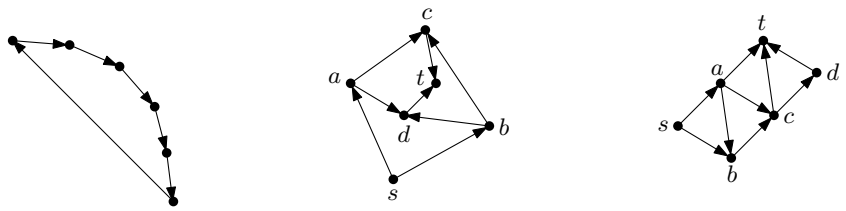
The intuition of this result is the following: The additional restriction that in a weak dominance drawing the successors have to be placed to the right and above their successors is only of geometric nature when compared to an upward drawing, where it is sufficient to place them above their predecessors.

Consider an upward planar drawing Γ and some point $p \in \Gamma$. By definition, all other points q for which there exists $p \prec q$, have to be placed above p. We may refer in an informal manner to the upper half plane as the *feasible drawing area* for successors in an upward drawing. When now considering a weak dominance drawing, then the feasible drawing area is the upper right quadrant. In the proof of Lemma 2.2, we essentially described an affine transformation that maps one feasible drawing area to the other. And as a result, we may state that restricting the feasible drawing area for successors does not necessarily restrict the class of graphs that admit such a drawing. Let us investigate this idea a bit further.

Di Giacomo et al. [40] introduce another form of drawing that in contrast to weak dominance drawings does not require a successor to be placed above and to the right of a vertex. Instead, it can be placed above or to the right. To avoid confusion, we introduce this drawing style under a slightly different name than in [40]. We say a planar straight-line drawing Γ of digraph G is a **weak upward-rightward (weak UR)** drawing, if for every pair of vertices $u, v \in V$ the following holds:

$$u \rightsquigarrow v \quad \Rightarrow \quad x(u) < x(v) \ \vee \ y(u) < y(v).$$

In [40] the authors show that every planar digraph admits such a drawing within polynomial area. Figure 2.6a shows a weak UR drawing of a simple cycle. A



(a) (b) (c)

Figure 2.6. (a) Weak UR drawing of a cycle that is not a strong UR drawing. (b) Weak UR drawing of the *st*-graph from Figure 2.4c that is not upward planar. (c) Strong UR drawing of the upward planar graph from Figure 2.4a.

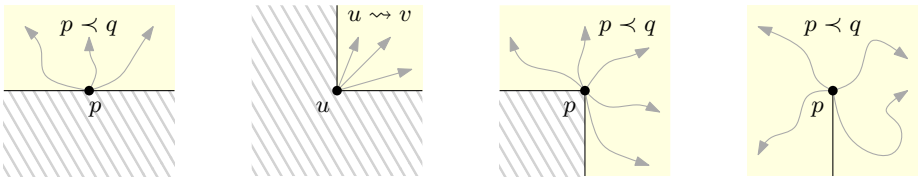
weak UR drawing of another graph that is not upward planar is depicted in Figure 2.6b. Notice that the feasible drawing area of weak UR drawings contains the upper half plane and the lower right quadrant.

During the conference at which the authors presented their results, a fruitful discussion emerged, and the question was asked, how a slightly stronger variant of their model influences the class of graphs that admit such a drawing. We introduce this model here and call it a **strong upward-rightward (strong UR)** drawing. More specifically, a planar drawing Γ of a digraph $G = (V, E)$ is called strong upward-rightward, if for every two points $p, q \in \Gamma$ it holds that

$$p \prec q \quad \Rightarrow \quad x(p) < x(q) \ \vee \ y(p) < y(q).$$

Notice that the only differences between weak and strong UR drawings are that the former requires edges to be straight-lines and constrains the vertex position, whereas for the later the requirement must hold for all points. Furthermore, a strong UR drawing is not necessarily a straight-line drawing. It is not difficult to see from their definitions that every strong UR-drawing is also a weak UR-drawing, but not necessarily the other way around as illustrated in Figure 2.6a. Moreover, it is not hard to see that every upward planar drawing is a strong UR-drawing. However, in comparison to an upward planar drawing, a strong UR-drawing offers more possibilities due to a larger feasible drawing area.

This observation gives rise to the following questions: Is there a digraph that admits a strong UR-drawing, but that is not upward planar? In other words: the strong UR-model is a relaxation of upward planar drawings from a geometric point of view, so can we draw more graphs in this model due to this relaxation? Compared to the relation of weak dominance drawings and upward planar graphs, where restricting the feasible drawing area of successors



(a) upward (b) weak-dom. (c) strong UR (d) ND

Figure 2.7. Comparison of the feasible drawing area for successors in the various models: (a) upward planar, (b) weak-dominance straight-line, (c) upward-rightward, and (d) not-downward. The striped area indicates forbidden locations for successors.

does not lead to a reduced class of graphs that admit such a drawing, we now ask the opposite question: Does enlarging the feasible drawing area result in a larger class of graphs?

In order to deal with these questions in more detail, we do not consider strong UR-drawings. Instead we introduce a new class of drawing that takes the idea of increasing the feasible drawing area to the extreme. Roughly speaking, we only require the upward condition, if two points have the same x-coordinate. This new type of drawing, we refer to as **not-downward (ND)** drawing, can be formalized as follows: A planar drawing Γ is an ND-drawing of a digraph G, if for every pair of points $p, q \in \Gamma$ the following holds:

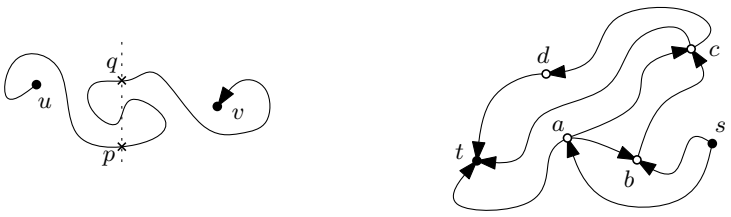
$$p \prec q \quad \Rightarrow \quad y(p) < y(q) \ \vee \ x(p) \neq x(q).$$

Figure 2.7 illustrates the feasible drawing areas of four drawing types including ND-drawings. By their definition, it follows immediately that every weak dominance, upward planar or strong UR-drawing is an ND-drawing. An example for the degree of freedom granted by this type of drawing is illustrated in Figure 2.8a, in which the curve of a single edge (u, v) that complies with the not-downward property is shown. An ND-drawing of a small planar st-graph is depicted in Figure 2.8b. Clearly, such a drawing is far from being an upward planar one.

Intuitively, one may think that the class of graphs that admit such an ND-drawing is much larger than the upward planar graphs. Surprisingly, this is not the case, and in the following, we show that these classes coincide, that is, the graphs admitting an ND-drawing are exactly the upward planar graphs.

We start with two simple lemmas to prepare for the main theorem.

▶ **Lemma 2.3.** *Let $G = (V, E)$ be a digraph and $G' = (V', E')$ a digraph that is obtained by replacing every edge $e = (u, v) \in E$ by a directed path $u \rightsquigarrow v$ of arbitrary length. If G' is upward planar, then G is upward planar.*



(a) (b)

Figure 2.8. (a) Example for an edge (u, v) in an ND-drawing. For every point pair p, q with $p \prec q$ and $x(p) = x(q)$, $y(p) < y(q)$ holds. (b) Example for an ND-drawing of the planar st-graph from Figure 2.4a.

Proof. By Theorem 2.1, G' admits an upward planar straight-line drawing. Then every edge $e = (u, v) \in E$ of G that has been replaced by a directed path $u \rightsquigarrow v$ in G' can be drawn as a poly-line using the inner vertices as bends. □

Although the ND-property provides many possibilities for drawing a digraph, the representation of cycles, however, is not one of them.

▶ **Lemma 2.4.** *A digraph that admits an ND-drawing is acyclic.*

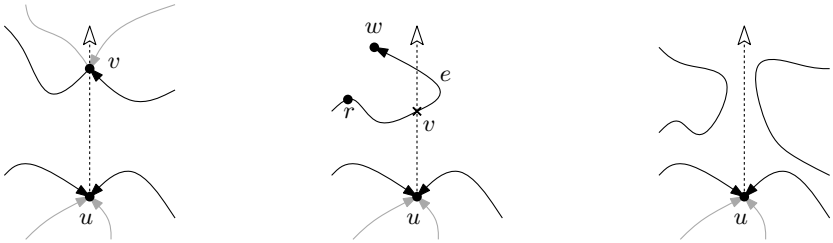
Proof. Assume to the contrary that there exists a cycle C. This cycle induces a closed jordan curve in any ND-drawing. There exists a vertical line that is crossed by C in at least two points $p \neq q$ such that $p \prec q$ and $q \prec p$ holds. Regardless of their exact y-coordinates, either $y(p) < y(q)$ or $y(p) > y(q)$ holds. Hence, in one of the two cases, the not-downward property is violated. □

We will exploit this property in the upcoming main theorem using a simple idea. If we augment a graph that admits an ND-drawing and are able to maintain a feasible ND-drawing, then we can claim that the resulting graph is acyclic. So instead of arguing along the structural changes made to prove that the result is acyclic, we argue with the existence of a feasible ND-drawing.

▶ **Theorem 2.5.** *A digraph admits an ND-drawing if and only if it is upward planar.*

Proof. By definition, every upward planar drawing is planar not-downward. Hence, it remains to show that a digraph $G = (V, E)$ that admits a not-downward planar drawing Γ is upward planar.

We first outline the overall idea of the proof. Given an ND-drawing for G, we augment G by splitting some edges to obtain a new digraph G'. G' has the property that every sink is on the outer face or has a vertex right above it.



(a) (b) (c)

Figure 2.9. The three possible cases when shooting a ray upwards from a sink u: the ray hits (a) a vertex v, (b) an edge $e = (r, w)$ or (c) nothing at all.

Similarly, every source has a vertex below or is accessible from the outer face. Adding edges between sinks (sources) and the corresponding vertices above (below) eliminates these sinks (sources), and enables us to show that G' is a subgraph of a planar st-graph G_{st}, which suffices for G being upward planar.

We start with describing the construction of G'. Let u be a sink in G. When shooting a ray from u upwards, three cases are possible (see Figure 2.9): The ray hits (i) a vertex v, (ii) an edge $e = (r, w)$ or (iii) nothing. The new digraph G' should only contain sinks for which either case (i) or (iii) applies. In case (ii), that is, in which we hit an edge $e = (r, w)$, we split e into two new edges $(r, v), (v, w)$ with v being a dummy vertex. In the drawing, we place v on the curve representing e such that $x(v) = x(u)$. The two new edges $(r, v), (v, w)$ inherit their layout from e. This way the result remains a feasible ND-drawing. For the sources in G, we proceed similarly by shooting a ray downwards.

Let the result of all these edge splits be $G' = (V', E')$. By construction, for every sink u, there either exists $v \in V'$ with $x(u) = x(v)$ and $y(u) < y(v)$, or the vertical ray starting from u going upwards does not intersect the drawing. In a symmetric manner, for every source v, there either exists $u \in V'$ with $x(u) = x(v)$ and $y(u) < y(v)$, or the vertical ray pointing downwards from v hits nothing. Moreover, the drawing Γ' is a feasible ND-drawing for G'.

As a next step, we show that when adding an edge (u, v) with u being a sink and v the corresponding vertex above, the result is still a feasible ND-drawing. The edge is drawn as a vertical segment from u to v. We only consider here the case in which u is a sink. For a source, the same procedure can be applied. We prove by induction over the number of edges inserted, say k, assuming that they are given in some arbitrary order. Let G'_k be the graph after adding the k-th edge, and Γ'_k the corresponding drawing. Clearly, $\Gamma'_0 = \Gamma'$ is a feasible ND-drawing for $G'_0 = G'$.

Now assume to the contrary that after adding the $k + 1$-th edge $e = (u, v)$,

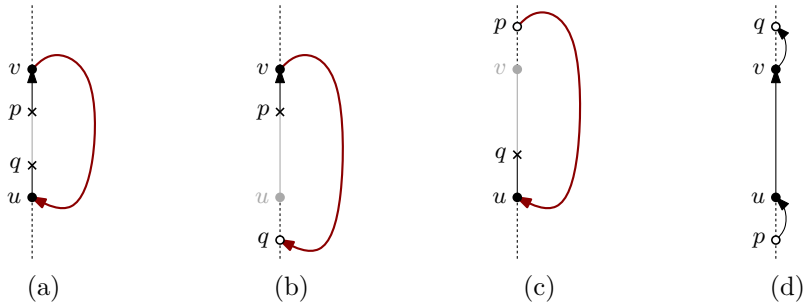


Figure 2.10. The four subcases to consider when p, q, u, v are vertically aligned. Bold curves represent contradicting paths. Gray parts of $e = (u, v)$ are not part of $p \rightsquigarrow q$. (a) p, q are both on $e = (u, v)$, (b) only p is on e, or (c) only q is on e. (d) Neither p nor q are on e implying $y(p) < y(q)$

Γ'_{k+1} is no longer a feasible ND-drawing for $G'_{k+1} = G'_k \cup e$. More specifically, if we add e to G'_k drawn as a vertical segment, then there exist two points p, q on a curve induced by a path such that $p \prec q$ and $x(p) = x(q) \wedge y(p) > y(q)$ holds. Furthermore, by our hypothesis, we may assume that Γ'_k is a feasible ND-drawing for G'_k, thus, this curve does not exist in Γ'_k. Clearly, the new edge $e = (u, v)$ must be involved in some way. Let l_{pq} and l_{uv} be the vertical lines through p, q and u, v, respectively. With a slight abuse of notation, we may refer with l_{pq} and l_{uv} to the subset of points or the x-coordinate of the vertical lines. Based on the relative position of l_{pq} and l_{uv}, we distinguish three main cases.

First, suppose that p, q, u and v are vertically aligned, that is, $l_{pq} = l_{uv}$. We show by enumeration of all subcases that p, q with $y(p) > y(q)$ cannot exist in Γ'_{k+1}, when Γ'_k is a feasible ND-drawing. In total there are four subcases to consider that depend on the relative position of $e = (u, v)$ and p, q. These can be summarized as follows: p, q lie on e (including u and v), either p or q is on e, or none of the two is on e. In order to simplify notation, we refer to p, q as vertices to prevent an excessive distinction between paths and edge segments. Notice that p, q are not necessarily vertices, but we may split an edge temporarily to achieve this. The feasibility of an ND-drawing is not affected, since it depends only on the curves induced by paths.

In the case p, q are both on e, it follows from $y(p) < y(q)$, that there exists a path $p \to v \rightsquigarrow u \to q$ in G'_{k+1} (Figure 2.10a). The subpath $v \rightsquigarrow u$ cannot contain (u, v), thus, exists in G'_k, but by construction $y(u) < y(v)$ holds, which contradicts that Γ'_k is a feasible ND-drawing. If only p is part of e but q is not, then there exists a path $p \to v \rightsquigarrow q$ with $v \rightsquigarrow q \in G'_k$ and $y(q) < y(v)$, a contradiction (Figure 2.10b). In a symmetric manner, that is, if q is on e instead of p, there exists $p \rightsquigarrow u \to q$, which implies $p \rightsquigarrow u \in G'_k$ and $y(u) < y(p)$,

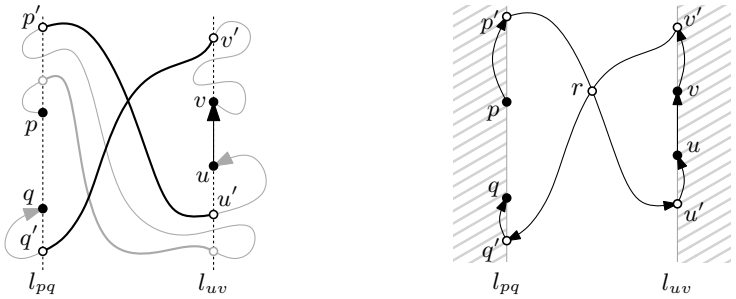


Figure 2.11. (a) Example layout of the path $p \rightsquigarrow u \to v \rightsquigarrow q$ in the case $l_{pq} < l_{uv}$. Circles indicate possible candidates for p', q', u' and v'. (b) Subpaths $p' \rightsquigarrow u'$ and $v' \rightsquigarrow q'$, both contained in the strip bounded by l_{pq} and l_{uv} with a common vertex r.

again a contradiction (Figure 2.10c). If neither p nor q are part of e, then e is completely involved, that is there exists $p \rightsquigarrow u \to v \rightsquigarrow q$ with $p \rightsquigarrow u \in G'_k$ and $v \rightsquigarrow q \in G'_k$, which yields $y(p) < y(u)$ and $y(v) < y(q)$. Since by construction $y(u) < y(v)$ holds, we get $y(p) < y(q)$, contradicting the assumption that $y(p) > y(q)$ (Figure 2.10d).

Now assume that $l_{pq} < l_{uv}$. Since neither p nor q can lie on $e = (u, v)$, there exists then a path $p \rightsquigarrow u \to v \rightsquigarrow q$. Let us take a closer look at the path $p \rightsquigarrow u$, more specifically, its intersection with l_{pq} and l_{uv}. There exists at least one such intersection on each side, namely at p and u themselves. However, there might exist several more (crossings or intersections). See Figure 2.11a for an example. Let p' be such a point of $(p \rightsquigarrow u) \cap l_{pq}$ (possibly p itself). Since this path does not contain (u, v) and $x(p) = x(p')$ holds, we may argue that in case $p \neq p'$, $p \rightsquigarrow p'$ also exists in G'_k, and, therefore, for every such p', $y(p) \leq y(p')$ holds. With the same argument, for every $u' \in (p \rightsquigarrow u) \cap l_{uv}$, it holds that $y(u') \leq y(u)$. Now, we choose p', u' such that their corresponding path $p' \rightsquigarrow u'$ has no other points in common with l_{pq}, l_{uv}, that is $(p' \rightsquigarrow u') \cap l_{pq} = \{p'\}$ and $(p' \rightsquigarrow u') \cap l_{uv} = \{u'\}$ holds. In a symmetric manner, we choose v' and q' such that $(v' \rightsquigarrow q') \cap l_{pq} = \{q'\}$ and $(v' \rightsquigarrow q') \cap l_{uv} = \{v'\}$ holds.

Hence, $p' \rightsquigarrow u'$ and $v' \rightsquigarrow q'$ are contained in the strip bounded by l_{pq} and l_{uv}, having only their endpoints in common with the two vertical lines (Figure 2.11b). More specifically, for every $r \in p' \rightsquigarrow u'$ with $p' \prec r \prec u'$, it holds that $l_{pq} < x(r) < l_{uv}$, and for every $r \in v' \rightsquigarrow q'$ with $v' \prec r \prec q'$, it holds that $l_{pq} < x(r) < l_{uv}$. Moreover, by construction, $y(u) < y(v)$, and, by assumption, $y(q) < y(p)$ holds, which yields $y(q') \leq y(q) < y(p) \leq y(p')$ and $y(u') \leq y(u) < y(v) \leq y(v')$.

Now consider Figure 2.11b in which the situation is illustrated. From $y(q') < y(p')$, $y(u') < y(v')$, and that $p' \rightsquigarrow u'$, $v' \rightsquigarrow q'$ are contained in the strip bounded by l_{pq}, l_{uv}, it follows that they cross or have a vertex in common. The former case contradicts planarity, so assume they share a vertex, say r with $p' \rightsquigarrow r \rightsquigarrow u'$ and $v' \rightsquigarrow r \rightsquigarrow q'$. Then there also exists $p' \rightsquigarrow r \rightsquigarrow q'$ and $v' \rightsquigarrow r \rightsquigarrow u'$. Notice that the newly added edge $e = (u, v)$ is not involved. Therefore, both paths exist in G'_k and contradict with $y(q') < y(p')$ and $y(u') < y(v')$ that Γ'_k is a feasible ND-drawing.

The remaining case $l_{pq} > l_{uv}$ is symmetric. One may just temporarily flip the drawing. We conclude that inserting $e = (u, v)$ drawn as a vertical segment, does not create a pair of points p, q such that $p \prec q$ with $x(p) = x(q)$ and $y(p) > y(q)$ holds. Therefore, if Γ'_k is a feasible ND-drawing, so is Γ'_{k+1} then. Hence, we can add all edges such that for the sinks and sources that remain, only case (iii) applies, that is, the corresponding sink or source is on the outer face.

In order to eliminate those, we add a super source s and super sink t that are placed below and above the drawings bounding box, respectively. Now we exploit that the vertical ray at a sink u hits nothing during the construction of G'. We insert an edge (u, t) using a vertical line segment until we hit the bounding box, then we turn towards t with a bend. For a source v, we proceed similarly by inserting an edge (s, u). Let the resulting graph be G_{st} for which we now claim that G_{st} is st-planar. Clearly, s and t are on the outer face. Moreover, it is not hard to see that G_{st} is planar and acyclic. This follows from the fact that $G_{st} - \{s, t\}$ has a feasible ND-drawing, thus, by Lemma 2.4 is acyclic. Addition of s and t does not induce additional cycles nor does it destroy planarity. By Theorem 2.1, G_{st} is upward planar. Moreover, $G_{st} - \{s, t\}$ is a subgraph of G_{st} and so is G' and, therefore, upward planar as well. From Lemma 2.3, it follows then that G is upward planar which concludes the proof. □

Since no algorithm has been described to actually create an ND-drawing, one may argue that it is an artificial construct. But this type of drawing provides a high degree of freedom by its definition such that it includes most of the other drawing classes presented so far. At the same time, Theorem 2.5 states that this relaxation does not lead to an enlarged class of graphs that admit such a drawing. Therefore, it may serve as an alternative geometric characterization of upward planarity. One may also say that the upward planar graphs are not only special in the sense that they admit an upward planar drawing, but that upward planarity itself is a requirement for a family of drawing styles. Figure 2.12 illustrates the landscape of the different drawing types, graph classes and their relations that have been discussed. Notice the cycle obtained from the proof of Theorem 2.5.

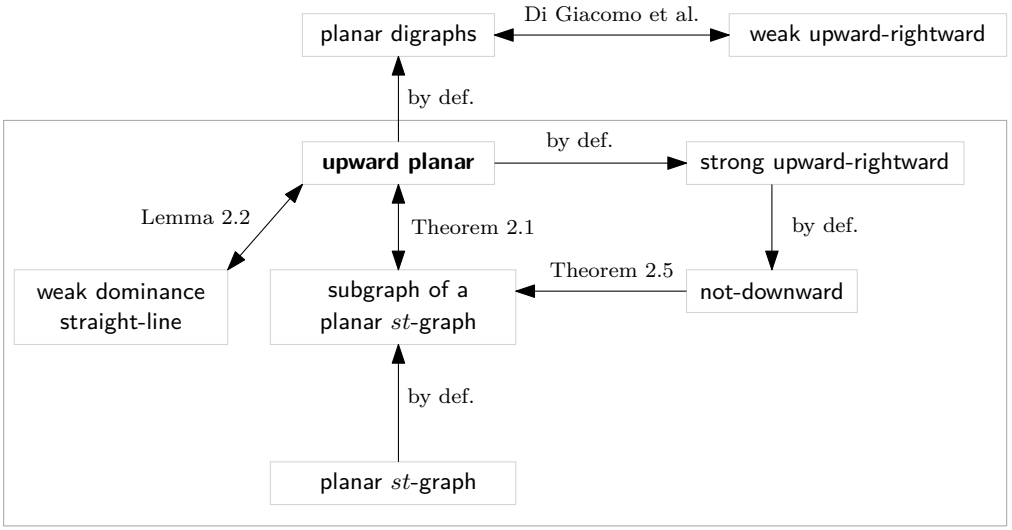


Figure 2.12. Discussed classes of planar digraphs, their drawings and relationships. An arc corresponds to the subset relation, for example, every planar st-graph is a subgraph of a planar st-graph, but not vice versa.

3 Bitonic st-orderings

As fundamental ingredients of incremental drawing procedures, various types of orderings have been developed and improved over the years. De Fraysseix, Pach and Pollack [34] introduced the canonical ordering to create straight-line drawings of maximal planar graphs. Afterwards, Kant [62] extended this concept to the triconnected case. Obtainable in linear-time, both have been used in the graph drawing literature extensively [5, 9, 22, 27, 44, 45, 62]. A few attempts have been made to generalize them to the biconnected case by relaxing their properties [51, 53]. However, an alternative that in nature works for biconnected graphs and that can be computed in linear time, are *st*-orderings [20, 47]. In the field of graph drawing, they have been used in several methods, reaching from the construction of visibility representations to drawings of non-planar graphs, see e.g. [16, 35, 66]. Although canonical and *st*-orderings share some properties in the planar case, it seems that they are usually not used in the same context.

In the following, we investigate these differences in more detail, especially one property of canonical orderings that is used implicitly in many drawing algorithms. Consider the successors of a single vertex in the clockwise ordering as implied by the embedding. Then their ranks in the canonical ordering form an increasing and then decreasing sequence, that is, a *bitonic* sequence. Common *st*-orderings do not necessarily have this property, rendering them unsuitable for some applications.

We counteract by introducing a new type of *st*-ordering for biconnected planar graphs: the *bitonic st-ordering*, an *st*-ordering in which the successors of every vertex appear in the aforementioned pattern. We show that every biconnected planar graph admits such an ordering. The proof is constructive and yields a linear-time algorithm that computes the ordering and a corresponding embedding. For the case where a fixed embedding is given, we prove that one cannot always find a bitonic *st*-ordering. In order to further support our idea, we describe two applications. In the first one, we extend the straight-line algorithm of de Fraysseix, Pach and Pollack [34] to bitonic *st*-orderings. In the second one, we describe how to obtain a special visibility representation and then transform it into a rectilinear T-shaped polygon contact representation.

Although the adaptation of the algorithm of de Fraysseix et al. does not yield any new results at first glance, it serves as an important algorithmic tool

in the second part, that is, the investigation of bitonic *st*-orderings for planar *st*-graphs. Unlike canonical orderings, the concept of *st*-orderings directly translates to directed graphs. Motivated by the observation that the presented straight-line algorithm creates an upward planar straight-line drawing from a bitonic *st*-ordering, we completely characterize the class of planar *st*-graphs for which a bitonic *st*-ordering exist. The result is a linear-time algorithm that recognizes these graphs and in case such an ordering exists, computes it within the same time bound.

For planar *st*-graphs that do not admit a bitonic *st*-ordering, we offer a different approach. We show that by splitting specific edges, we can enforce the existence of such an ordering. In general one would like to split as few edges as possible, since, for example, in our application every additional vertex corresponds to a bend in the resulting upward drawing. With this in mind, we describe a linear-time algorithm that computes the minimum set of edges to split such that the resulting graph admits a bitonic *st*-ordering. Although this does not necessarily minimize the number of bends in an upward planar drawing, we are able to improve upon the best known theoretical bound. More specifically, we show that every embedded planar *st*-graph has an upward planar poly-line drawing within quadratic area with at most $|V|-3$ bends in total and at most one bend per edge. Furthermore, such a drawing can be obtained in linear time.

3.1 Bitonic *st*-orderings of biconnected planar graphs

In the following, we first introduce some notations and definitions that are used throughout this chapter. If not stated otherwise, we consider in this section only simple planar connected graphs.

Assume that we are given a planar graph G embedded in the plane together with an *st*-ordering π for G such that s and t are on the outer face. π has a nice property which has been used in the graph drawing literature extensively [35]: When considering the circular ordering of the neighbors at a vertex v as implied by the embedding, then the set of predecessors and successors of v with respect to π form a consecutive sequence. We denote the ordered sequence of successors of a vertex v by $S(v) = \{w_1, \ldots, w_m\}$ such that for $1 \leq i < m$, w_i precedes w_{i+1} in the circular clockwise ordering around v and $\pi(v) < \pi(w_i)$ holds for all $1 \leq i \leq m$. This is well-defined for every $v \neq s$ due to the presence of at least one predecessor. For $S(s)$ with $|S(s)| > 1$, we choose w_1 to be the vertex that follows s on the boundary of the outer face in clockwise direction.

The property that the predecessors and successors appear consecutively is

particularly useful in an incremental drawing procedure. However, one has no control over which successor is placed when. Consider a simple example where a vertex v has been placed that has three successors, let us say $S(v) = \{w_1, w_2, w_3\}$. Then, π may be chosen such that w_2 must be placed before w_1 and w_3, that is, $\pi(w_2) < \pi(w_1)$ and $\pi(w_2) < \pi(w_3)$ holds. This may cause problems when attaching the edges (v, w_1) and (v, w_3), since (v, w_2) has already been attached.

This lack of control is avoided by using canonical orderings. De Fraysseix et al. [34] were the first to introduce this concept by giving the following definition:

▶ **Definition 3.1** (Canonical ordering for maximal planar graphs [34]). *Let $G = (V, E)$ be a maximal planar graph embedded in the plane with exterior face u_1, u_2, u_3. An ordering $v_1 = u_1, v_2 = u_2 \ldots, v_n = u_3$ of the vertices is a canonical ordering if for every $4 \leq k \leq n$, the following holds:*

- *The subgraph $G_{k-1} \subset G$ induced by $v_1, v_2, \ldots v_{k-1}$ is biconnected, and the boundary of its exterior face is a cycle C_{k-1} containing the edge (v_1, v_2).*
- *v_k is in the exterior face of G_{k-1}, and its neighbors in G_{k-1} form an (at least 2-element) subinterval of the path $C_{k-1} - (v_1, v_2)$.*

A situation as in the small example cannot occur when using canonical orderings, because of the biconnectivity of G_k. In fact one can go one step further and claim (as we did in the introduction) that the partition indices of the successors when considered in the clockwise ordering as implied by the embedding, form an increasing and then decreasing sequence. We will prove this for canonical orderings as an intermediate step later. For now we refer to this as the bitonic property.

Kant [62] extended the idea of de Fraysseix et al. to triconnected planar graphs. His definition, however, is not based on a vertex ordering, instead an ordered partition of the vertices is used.

▶ **Definition 3.2** (Triconnected canonical ordering [62]). *Let $G = (V, E)$ be a triconnected plane graph and (v_1, v_2) an edge on the outer face. Let $V_1 \cup \cdots \cup V_K$ be an ordered partition of V and G_k $(1 \leq k \leq K)$ the subgraph induced by $V_1 \cup \cdots \cup V_k$ with outer face C_k. $V_1 \cup \cdots \cup V_K$ is a canonical ordering of G if:*

- *$V_1 = \{v_1, v_2\}$ and $V_K = \{v_n\}$, where v_n lies on the outer face and is a neighbor of v_1.*
- *Each C_k $(k > 1)$ is a cycle containing (v_1, v_2).*
- *Each G_k is biconnected and internally triconnected.*

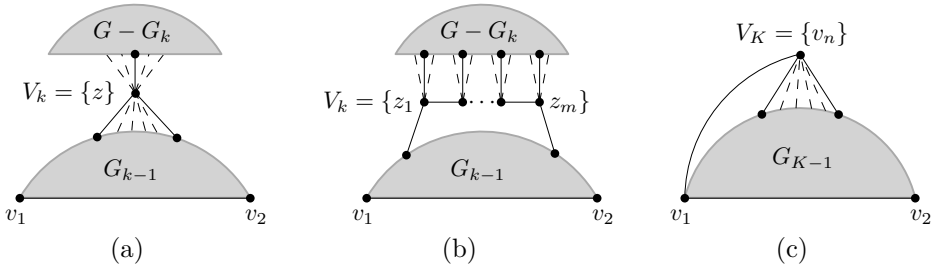


Figure 3.1. Triconnected canonical ordering as defined by Kant [62]: (a) A singleton $V_k = \{z\}$ with at least two neighbors in G_{k-1} and at least one in $G - G_{k-1}$. Required edges are drawn solid whereas optional edges are drawn dashed. (b) A chain $V_k = \{z_1, \ldots, z_m\}$ where z_1 and z_m have exactly one neighbor in G_{k-1}. (c) The last vertex v_n that is adjacent to v_1.

- *For $1 < k < K$ one of the two following conditions holds:*
  1. *$V_k = \{z\}$ is a singleton where z belongs to C_k and has at least one neighbor in $G - G_k$.*
  2. *$V_k = \{z_1, \ldots, z_m\}$ where each z_i $(1 \leq i \leq m)$ has at least one neighbor in $G - G_k$, and where z_1 and z_m each have one neighbor in C_{k-1}, and these are the only two neighbors of V_k in G_{k-1}.*

In difference to the definition given by de Fraysseix et al., Kant distinguishes between two types of partitions. The singletons, partitions consisting of a single vertex, play a similar role as in Definition 3.1 for maximal planar graphs (Figure 3.1a and 3.1c). However, chains, that are partitions consisting of more than one vertex, have a different structure since not all vertices have a neighbor in G_{k-1}. See Figure 3.1b for an illustration.

The concept of canonical ordering has been generalized to the biconnected case. Gutwenger and Mutzel [51] use an ordered partition of the vertices, referred to as *biconnected shelling ordering*, to create poly-line drawings in an incremental manner. A similar but more vertex ordering-based concept is used by Harel and Sardas [53]. They introduce the so called *biconnected canonical ordering* for drawing planar graphs in a straight-line style. In contrast to the triconnected variant where it is guaranteed that G_k is biconnected, one has now to deal with the additional problem that a vertex v_k may only have one neighbor in G_{k-1}. Harel and Sardas introduce for those vertices the property of having *left*, *right* and *legal support*.

▶ **Definition 3.3** (Left, right and legal support [53]). *Let G be a biconnected planar graph drawn in the plane, G_k be a connected subgraph of G, and $C_k =$*

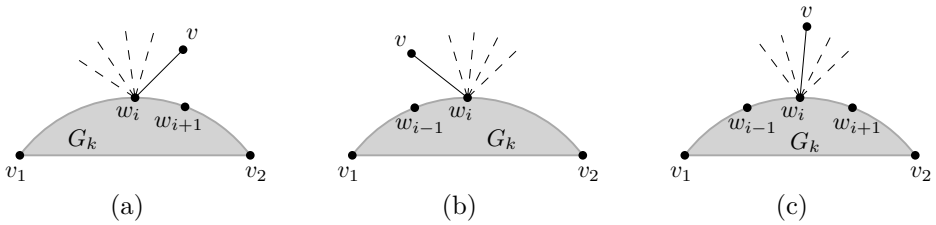


Figure 3.2. A vertex v that has only one neighbor in G_k. In (a) v has right support on the contour $C_k = \{v_1 = w_1, w_2, \ldots, w_m = v_2\}$. Possible other edges incident to w_i whose other endpoints are in $G - G_k$ are drawn dashed. In (b) a symmetric situation when v has left support, whereas in (c) v has neither left nor right support.

$w_1, w_2, \ldots, w_m$ be the counter-clockwise boundary list of the exterior face of G_k. Moreover, let $v \in G - G_k$ be a vertex that lies in the exterior face of G_k, and which has exactly one neighbor in G_k that by planarity must lie on C_k. Assume this neighbor is w_i for some i with $1 \leq i \leq m$.

- *We say that v has a right support if v immediately follows w_{i+1} in the counter clockwise circular ordering around w_i; it has a left support if v immediately precedes w_{i-1} in the counter clockwise circular ordering around w_i.*
- *We say that v has a legal support on C_k, if $i = 1$ and v has a right support, or $i = m$ and v has a left support, or $1 < i < m$ and v has a left support or a right support.*

The property of having either right or left support is shown in Figure 3.2a and Figure 3.2b, respectively. An example for a situation that is not allowed is given in Figure 3.2c where v follows and precedes two vertices in the counter clockwise circular ordering around w_i that are both not in G_k. This property is a fundamental requirement for a biconnected canonical ordering that is used in an incremental drawing procedure, because it may be used to establish the bitonic property as in the triconnected version. A similar mechanism can be found in the shelling ordering of Gutwenger and Mutzel [51]. In both definitions, the constraints of the triconnected version have been relaxed. Since we will use some more ideas from Harel and Sardas later, we give their definition here.

▶ **Definition 3.4** (Biconnected canonical ordering [53])**.** *Let G be a biconnected planar graph drawn in the plane, and let (u, v) be an edge on the clockwise boundary list of its exterior face. A biconnected canonical ordering is a labeling of the vertices of G in a sequence $v_1, \ldots, v_n$ such that $v_1 = u$ and $v_2 = v$, and for every $2 \leq k \leq n$ the following hold:*

- *Let G_k be the subgraph of G induced by $v_1, \ldots, v_k$. Then G_k is connected, and the edge (v_2, v_1) is on C_k, the contour of G_k. Fix w_1 to be v_1, so that we write C_k as $v_1 = w_1, w_2, \ldots, w_m = v_2$.*
- *All vertices of $G - G_k$ lie within the exterior face of G_k.*
- *For $k > 2$, the vertex v_k has one or more neighbors in G_{k-1}, If v_k has exactly one neighbor in G_{k-1}, then it has a legal support on C_{k-1}.*

This generalization sacrifices an important property that is required for some applications. In the triconnected case, every vertex $v \in V_k$, except for $k = K$, has a neighbor in $G - G_k$. Clearly, this property is missing in the biconnected canonical ordering of Harel and Sardas and we are not aware of any canonical ordering-like approach for the biconnected case, where this is guaranteed. In order to draw a connection to *st*-orderings, we refer to this property as the successor property. Table 3.1 summarizes the orderings and their features including our contribution (*bitonic st-ordering*).

	biconnected	successor	bitonic
st-ordering	yes	yes	no
biconnected shelling- & canonical ordering	yes	no	yes
canonical ordering	no	yes	yes
bitonic *st*-ordering	yes	yes	yes

Table 3.1. Comparison of the features of various orderings.

Another common technique for the biconnected case that can be found in the literature is to first develop an algorithm using the canonical ordering of Kant and is therefore limited to triconnected graphs. Afterwards, the algorithm is extended to the biconnected case using SPQR-trees. The main task can be sketched as follows. The original algorithm serves as a basis for the R-node case. It is then modified such that each (virtual) edge in the drawing can be replaced recursively by a drawing of the corresponding pertinent graph. Usually a drawing has to match certain invariant properties. For S- and P-nodes alternative methods are used. Finding a good invariant and presenting a clear proof can be tedious work and its complexity may outweigh the description of the original triconnected algorithm. We offer a different approach by defining a new type of *st*-ordering whose successor lists have the aforementioned property of being bitonic. A sequence is said to be *bitonic*, if it can be partitioned into two subsequences such that one is monotonically increasing while the other is decreasing. More specifically:

Figure 3.3. (a) Example in which seven successors of a vertex v are placed in a non-bitonic manner. The last three edges to be attached to v (dashed) are separated by previously attached ones (solid). In (b), when using a bitonic ordering, they appear consecutively in the embedding around v.

▶ **Definition 3.5.** *An ordered sequence $A = \{a_1, \ldots, a_n\}$ is **bitonic increasing**, if there exists $1 \leq k \leq n$ such that $a_1 \leq \cdots \leq a_k \geq \cdots \geq a_n$ holds and **bitonic decreasing** if $a_1 \geq \cdots \geq a_k \leq \cdots \leq a_n$. Moreover, we say A is bitonic increasing (decreasing) with respect to a function f if $A' = \{f(a_1), \ldots, f(a_n)\}$ is bitonic increasing (decreasing).*

One property of bitonic sequences that is very useful in our context is the following:

▶ **Property 3.6.** *If a sequence $A = \{a_1, \ldots, a_n\}$ is bitonic increasing (decreasing), then the reversed sequence $A' = \{a_n, \ldots, a_1\}$ is bitonic increasing (decreasing) as well.*

In the following, we restrict ourselves to bitonic increasing sequences. Thus, we abbreviate it by just referring to it as being bitonic.

▶ **Definition 3.7.** *Let $G = (V, E)$ be a planar graph embedded in the plane and $s, t \in V$ two vertices on the outer face. An st-ordering π is a **bitonic st-ordering** for G, if at every vertex $v \in V$ the ordered sequence of successors $S(v) = \{w_1, \ldots, w_m\}$ as implied by the embedding is bitonic with respect to π.*

Recall that not every graph admits an st-ordering. More specifically, a graph G admits an st-ordering if and only if $G \cup \{(s, t)\}$ is biconnected, see e.g. [20]. We therefore assume that this is the case. An st-ordering with the additional property of being bitonic is particularly useful in an incremental algorithm; the edges that correspond to those successors of a vertex v that have not been placed yet, appear consecutively in the embedding around v. See Figure 3.3 for an example.

Next, we describe how to obtain such a bitonic st-ordering.

3.1.1 A linear-time algorithm

The overall idea is to use the aforementioned SPQR-tree approach to derive a bitonic *st*-ordering instead of a drawing. Following the same steps, we first describe how to obtain a bitonic *st*-ordering for the triconnected case. This result then serves as a basis for the R-node case when extending the idea to the biconnected case using SPQR-trees.

▶ **Lemma 3.8.** *Every triconnected planar graph $G = (V, E)$ admits a bitonic st-ordering for every $(s, t) \in E$.*

Proof. From its definition it is easy to see that a triconnected canonical ordering $V_1 \cup \cdots \cup V_K$ as defined by Kant (see Definition 3.2) can be transformed into an *st*-ordering π. We start by describing the construction of π and then show that it is indeed bitonic with respect to π. Given an edge $(s, t) \in E$, we compute a canonical ordering $V_1 \cup \cdots \cup V_K$ of G by choosing $V_1 = \{s, s'\}$ and $V_K = \{t\}$ with s' being the vertex that precedes t in the clockwise order around s. Notice that by definition of the canonical ordering, the edges (s, t) and (s, s') are on the outer face. For the *st*-ordering π we follow a simple principle that is sometimes referred to as the vertex ordering of a canonical ordering: Regardless of $V_k = \{v_1, \ldots, v_m\}$ with $1 \leq k \leq K$ being a chain or singleton, we choose π for $1 \leq i \leq m$ such that $\pi(v_i) = |V_1| + \cdots + |V_{k-1}| + i$.

For the sake of notation, we refer to the partition of a vertex $v \in V_k$ by $\pi'(v) = k$. Notice that, by construction of π, for all $u, v \in V$ with $\pi'(u) < \pi'(v)$, it holds that $\pi(u) < \pi(v)$. By definition of the canonical ordering, every $v \in V_k$ with $k < K$ has at least one neighbor w in $V_{k+1} \cup \cdots \cup V_K$. It holds then that $\pi(w) > \pi(v)$ and as a result every $v \neq t$ has at least one successor. In case $V_k = \{v\}$ $(1 < k \leq K)$ is a singleton, v has at least two neighbors, say c_l and c_r, in $V_1 \cup \cdots \cup V_{k-1}$ with $\pi(c_l) < \pi(v)$ and $\pi(c_r) < \pi(v)$, thus v has at least two predecessors. In the other case, that is, $V_k = \{v_1, \ldots, v_m\}$ $(k > 1)$ is a chain, only v_1 and v_m have one neighbor each, let us say c_l and c_r, in $V_1 \cup \cdots \cup V_{k-1}$. However, for every $v_i \in V_k$ with $i > 1$ it holds that $\pi(v_{i-1}) < \pi(v_i)$. Hence, every v_i with $i < m$ has exactly one predecessor while v_m has even two. Special attention must be paid to $V_1 = \{s, s'\}$ since for this chain no c_l and c_r exist. However, the predecessor of s' is s and s itself does not require a predecessor for π being an *st*-ordering. Since all vertices $v \neq s$ have predecessors the order in $S(v)$ is well-defined by considering them clockwise. For s we have to break the cyclic order and set $S(s) = \{t = w_1, w_2, \ldots, w_{m-1}, w_m = s'\}$.

In order to prove that π is a bitonic *st*-ordering, we first show that every successor list obtained from π is bitonic with respect to π' instead of π. To do so, assume to the contrary that there exists a successor list $S(v) =$

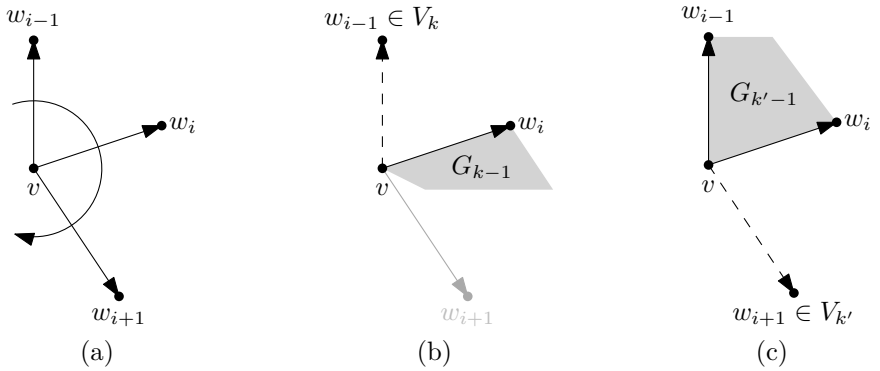


Figure 3.4. (a) The initial situation at v with $S(v) = \{\ldots, w_{i-1}, w_i, w_{i+1}, \ldots\}$. (b) G_{k-1} with $k = \pi'(w_{i-1})$ where w_{i-1} has to be in the outer face of G_{k-1}. (c) $G_{k'-1}$ with $k < k' = \pi'(w_{i-1})$ where w_{i+1} has to be in the outer face of $G_{k'-1}$.

$\{w_1, \ldots, w_i, \ldots, w_m\}$ of some vertex v that is not bitonic with respect to π', that is, there is a $w_i \in S(v)$ with $1 < i < m$ for which $\pi'(w_{i-1}) > \pi'(w_i)$ and $\pi'(w_{i+1}) > \pi'(w_i)$ holds. Furthermore, let w.l.o.g. $\pi'(w_{i-1}) < \pi'(w_{i+1})$. Notice that by construction of π and $S(v)$, it follows that $\pi'(w_{i-1}) \neq \pi'(w_{i+1})$. See Figure 3.4a for the initial situation at v. Now we set $k = \pi'(w_{i-1})$ and $k' = \pi'(w_{i+1})$ and argue that in a canonical ordering this can only occur for $k = 2$. By definition of the canonical ordering, $w_{i-1} \in V_k$ has to be in the outer face of G_{k-1} as displayed in Figure 3.4b. Similarly, $w_{i+1} \in V_{k'}$ has to be in the outer face of $G_{k'-1}$ (see Figure 3.4c). As a result, the outer face of G_{k-1} must be on both sides of the edge (v, w_i) and there is only one such G_{k-1} for which this is the case, namely G_1. Hence, $k = 2$, $v = s$, $w_i = s'$ and $w_{i+1} = t$. However, we defined $S(s)$ such that it ends with $w_m = s'$ which is a contradiction.

It remains to show that all $S(v)$ are not only bitonic with respect to π', but also for π. As aforementioned, by construction of π from π', for two vertices $u, v \in V$ with $\pi'(u) < \pi'(v)$ it follows that $\pi(u) < \pi(v)$. And since we have just shown for the successor list $S(v) = \{w_1, \ldots, w_i, \ldots, w_m\}$ of every vertex $v \in V$ it holds that $\pi'(w_{i-1}) < \pi'(w_i)$ or $\pi'(w_{i+1}) < \pi'(w_i)$, we may deduce that $\pi(w_{i-1}) < \pi(w_i)$ or $\pi(w_{i+1}) < \pi(w_i)$. Hence, every $S(v)$ is bitonic with respect to π. □

The proof is constructive and reveals one additional property: The successor list of s is a special case, because it contains s' and t. Furthermore, s is the only vertex with $\pi(s) < \pi(s')$ and for every vertex $v \in V$ with $v \neq t$, $\pi(v) < \pi(t)$ holds. Since the successor list of s starts with t, ends with s' by our construction, and is bitonic with respect to π, we can state the following:

▶ **Corollary 3.9.** *The successor list of s starts with t, ends with s' and is sorted decreasingly with respect to π, that is, $S(s) = \{t, w_2, \ldots, w_{m-1}, s'\}$ such that $\pi(t) > \pi(w_2) > \cdots > \pi(w_{m-1}) > \pi(s')$.*

While the above results follow the intuition of canonical orderings, they hold only for the case where the input is triconnected. Next, we extend this result to the biconnected case using SPQR-trees. Corollary 3.9 provides us with the necessary ingredient for an invariant. More details are given in the proof of the main result of this section:

▶ **Theorem 3.10.** *Every biconnected planar graph $G = (V, E)$ has a bitonic st-ordering π for any given st-edge $e^* \in E$. The ordering π and a corresponding embedding can be computed in time $\mathcal{O}(|V|)$.*

Proof. The overall challenge is to recursively compose a bitonic st-ordering along an SPQR-tree. For a subtree, we assume that we have already constructed a bitonic st-ordering that complies with an invariant. Then we show that we can combine it with the solutions of other subtrees in the skeleton of the parent node.

Invariant: For the assignment of an index in π, we maintain a single global counter that we use to label the vertices in an incremental manner. The poles $\{s, t\}$ of a tree node μ are labeled by the parent. Moreover, s has already been labeled such that we may assume that the global counter has a value greater than $\pi(s)$. Furthermore, π is a bitonic st-ordering for the subgraph induced by μ when assigning t the current value of the counter. Additionally, the successor list of s is sorted decreasingly with respect to π. We start by embedding G, creating the SPQR-tree $\mathcal{T}$ and rooting it at the Q-node representing the given st-edge $e^* = (s^*, t^*)$. Then we initialize the global counter, label s^*, and recurse on the only child of the root. Following standard practice, we now distinguish the different types of tree nodes.

Serial case: Let μ be an S-node whose skeleton consists of the simple cycle $s, v_1, \ldots, v_{m-1}, t, s$, with (s, t) being the reference edge representing the parent of μ. The remaining edges $(s, v_1), \ldots, (v_{m-1}, t)$ correspond to the children $\mu_1, \ldots, \mu_m$ of μ. We recurse on μ_1, label v_1, recurse on μ_2, and so on, until μ_m. Notice that we do not label t. Clearly, the result is an st-ordering when assigning t the current value of the counter. The successor lists of $s, v_1, \ldots, v_{m-1}$ are all sorted decreasingly due to our invariant, thus, are bitonic.

Parallel case: We first check if one of the children $\mu_1, \ldots, \mu_m$ of the P-node μ is a Q-node. In that case we change the order of the children such that μ_1 is the Q-node. Notice that this implies a change in the embedding of G. Then we

recurse on the children in their reverse order, that is $\mu_m, \ldots, \mu_1$. Consider now the successor list $S(s)$ of s: The neighbors $w_1^i, \ldots, w_{k'}^i$ with $1 \leq i \leq m$ that are located in the induced subgraph of μ_i form a consecutive sequence in $S(s)$:

$$S(s) = \{\ldots, \underbrace{w_1^i, \ldots, w_k^i}_{\text{neighbors in } \mu_i}, \underbrace{w_1^{i+1}, \ldots, w_{k'}^{i+1}}_{\text{neighbors in } \mu_{i+1}}, \ldots\}$$

By our invariant, it follows that $\pi(w_j^i) > \pi(w_{j+1}^i)$ and since we recursed on $\mu_1, \ldots, \mu_m$ in reverse order, $\pi(w_k^i) > \pi(w_1^{i+1})$ holds. Hence, the sequence is decreasing.

Rigid case: We start by constructing a temporary bitonic st-ordering π' for the triconnected skeleton $G_\mu = (V_\mu, E_\mu)$ of the R-node μ using Lemma 3.8 and choosing the reference edge (s, t) as input. Then we traverse the vertices of V_μ in the ordering as given by π'. At a vertex $v \in V_\mu$, we recurse on the incident edges $(u, v) \in E_\mu$ with $\pi'(u) < \pi'(v)$, that is, the incoming edges of v with respect to π'. Afterwards, we label v unless $v = t$. The resulting ordering is not necessarily a bitonic st-ordering. We proceed in two steps: First we derive some useful properties of π and narrow down the problem. Then we argue that mirroring the embedding of some children of μ changes the successor lists such that they become bitonic with respect to π.

Let us take a closer look at the properties of π: Since we labeled all $v \in V_\mu$ in the order as provided by π', for any two vertices $u, v \in V_\mu$ with $u \neq v$, it holds that $\pi'(u) < \pi'(v)$ if and only if $\pi(u) < \pi(v)$. Hence, π is a feasible bitonic st-ordering for G_μ. Recall that we recursed on the children in a special way. Consider a vertex v' in the induced subgraph of a child μ_{uv} represented by the virtual edge $(u, v) \in E_\mu$ with $\pi(u) < \pi(v)$. Furthermore, assume that v' is not a pole of μ_{uv}, that is, $u \neq v' \neq v$. Then v' has been labeled before v and after any $w \in V_\mu$ with $\pi(w) < \pi(v)$, thus $\pi(w) < \pi(v') < \pi(v)$. When now considering a fourth vertex, say w', that is defined similar as v', that is, a non-pole vertex located in the subgraph induced by a virtual edge $(x, w) \in E_\mu$ with $\pi(x) < \pi(w)$, then we may deduce the implication $\pi(w) < \pi(v) \Rightarrow \pi(w') < \pi(v')$. Stemming from the special traversal of the edges, this property is of particular interest when considering the successor lists.

Let $S'(v) = \{w'_1, \ldots, w'_h, \ldots, w'_m\} \subset V_\mu$ be the successor list of $v \in V_\mu$, for which an example is shown in Figure 3.5a. Notice that $\pi(w'_1) < \cdots < \pi(w'_h) > \cdots > \pi(w'_m)$ holds. Furthermore, let $\mu_1, \ldots, \mu_m$ be the corresponding children of μ that are represented by the virtual edges $(v, w'_1), \ldots, (v, w'_m)$ with $\pi(v) < \pi(w'_i)$ for $1 \leq i \leq m$. Similar to the P-node case, we refer to the neighbors of v that are contained in the subgraph induced by μ_i as $w_1^i, \ldots, w_{k_i}^i$.

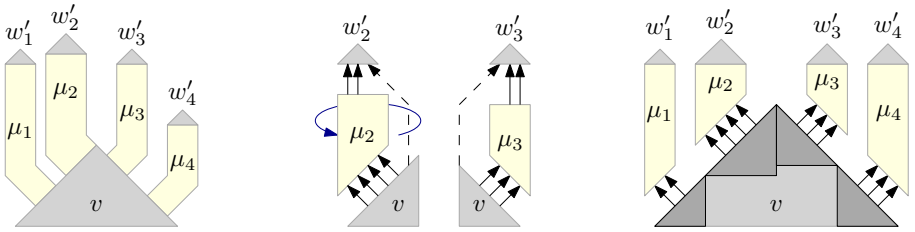


Figure 3.5. (a) Example of virtual edges $(v, w'_1), \ldots, (v, w'_4)$ in an R-node representing the tree nodes $\mu_1, \ldots, \mu_4$. (b) Mirroring the embedding of the subgraph induced by μ_2, effectively turning the decreasing sequence into an increasing sequence. (c) The bitonic successor list at v after mirroring the embedding of μ_1 and μ_2.

These form a consecutive sequence in $S(v)$, hence, we may write $S(v)$ as

$$S(v) = \{\underbrace{w_1^1, \ldots, w_{k_1}^1}_{\text{neighbors in } \mu_1}, \ \ldots, \ \underbrace{w_1^h, \ldots, w_{k_h}^h}_{\text{neighbors in } \mu_h}, \ \ldots, \ \underbrace{w_1^m, \ldots, w_{k_m}^m}_{\text{neighbors in } \mu_m}\}.$$

The idea now is to distinguish between two cases, depending on whether $i < h$ or $i \geq h$ holds, that is, w'_i is in either the increasing or decreasing partition of $S'(v)$.

Let us first consider the case $h \leq i$: Since $\pi(w'_i) > \pi(w'_{i+1})$ for $h \leq i < m$, it follows that $\pi(w_{k_i}^i) > \pi(w_1^{i+1})$ for all $h \leq i < m$, that is, the last neighbor in the subgraph induced by μ_i has a greater label than the one in μ_{i+1}. By our invariant we may assume that $\pi(w_1^i) > \cdots > \pi(w_{k_i}^i)$ for all $h \leq i \leq m$ holds, that is, with respect to π, we have a decreasing subsequence in $S(v)$. Hence, the sequence $w_1^h, \ldots, w_{k_m}^m$ is decreasing with respect to π.

In the second case where $1 \leq i < h$ holds, an increasing sequence is required. We mirror the embedding of every subgraph induced by μ_i with $1 \leq i < h$ along its poles (v, w'_i). As a result the decreasing subsequences in $S(v)$ turn into increasing ones, that is, $\pi(w_1^i) < \cdots < \pi(w_{k_i}^i)$ for all $1 \leq i < h$ (μ_2 in Figure 3.5b). Notice that by Property 3.6 the successor list of every vertex in the mirrored subgraph remains bitonic. Now similar to the first case, we argue that from $\pi(w'_i) < \pi(w'_{i+1})$ it follows that $\pi(w_{k_i}^i) < \pi(w_1^{i+1})$ for all $1 \leq i < h$. Thus, the sequence $w_1^1, \ldots, w_{k_{h-1}}^{h-1}$ is increasing with respect to π. And as a result, the sequence $w_1^1, \ldots, w_{k_{h-1}}^{h-1}, w_1^h, \ldots, w_{k_m}^m$ is bitonic with respect to π (Figure 3.5c). Notice that for $v = s$, there exists no i with $\pi(w'_i) < \pi(w'_{i+1})$, thus, $S(s)$ is sorted decreasingly with respect to π as required by the invariant.

The case where μ is a Q-node is trivial. It remains to label t at the root, which, by our invariant, results in a bitonic st-ordering. Both, the canonical ordering and the SPQR-tree, can be computed in linear time, thus, the runtime follows immediately. □

In the proof of the main theorem, we changed the embedding of G in two places. At first, in the P-node case, we had to ensure that a possible Q-node follows the reference edge in clockwise order around s. Afterwards, in the R-node case, we mirrored the embedding along the poles to turn a decreasing sequence into an increasing one. The latter change is caused by our invariant that only provides a decreasing sequence at s for the sake of an easier maintainable invariant. In an actual implementation, this can easily be avoided by mirroring the embedding twice, once before recursing on the corresponding child and then afterwards. Thus, the resulting embedding is equivalent to the initial one. However, the P-node case is not trivial and the question may arise if it is necessary in general, or if one may always find a bitonic st-ordering for every edge when a fixed embedding is given.

```
procedure labelMain(μ, isFlipped)
begin
    // Flip the embedding of the skeleton
    if isFlipped then
        flipEmbeddingAlongPoles(G_μ)
    end
    // Label it based on the type
    switch type of μ do
        case S-node: labelSnode(μ, isFlipped);
        case P-node: labelPnode(μ, isFlipped);
        case R-node: labelRnode(μ, isFlipped);
    endsw
    // Restore the initial embedding
    if isFlipped then
        flipEmbeddingAlongPoles(G_μ)
    end
end
```

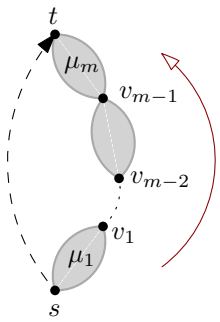
Algorithm 1: The main recursive labeling procedure.

Before we address this question, we describe the algorithm in more detail to outline the basic corner stones of an efficient implementation. To deal with the embedding changes required by the R-node case in an efficient manner, we have to avoid to flip the embedding of the complete subgraph induced by a tree

node μ. However, since the labeling procedure works solely on the skeletons, it is sufficient to consider their embedding and project it back on the graph afterwards. Moreover, we know in advance if a child has to be flipped after its subtree has been labeled.

The idea is to carry a boolean flag along when recursing on a subtree of a tree node μ. This flag indicates if such a change is required later by a parent of μ. Algorithm 1 outlines the idea of our main recursive function LABELMAIN that takes care of a possible flip along the poles. Before any labeling or recursion takes place, the function checks if the parent has set the flag *isFlipped* which indicates that the current skeleton G_μ has to be flipped afterwards such that the successor lists are being reversed. If so, it performs a first flip right away which does not affect any labels. Then the function distinguishes between the different types of tree nodes and calls the corresponding function. Afterwards, when the complete subtree has been labeled, the skeleton is flipped again, turning a decreasing sequence at s into an increasing one and restoring the original embedding at the same time.

```
procedure LABELSNODE(μ, isFlipped)
begin
    for i ← 1 to m do
        LABELMAIN(μ_i, isFlipped);
        if v_i ≠ t then
            π(v_i) ← cnt;
            cnt ← cnt + 1;
        end
    end
end
```

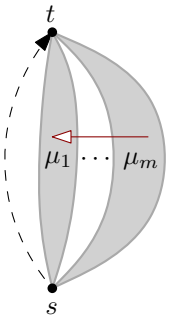


Algorithm 2: Labeling a chain in the S-node case.

When now considering for example the procedure LABELSNODE that handles the skeleton of an S-node (see Algorithm 2), the value of *isFlipped* is passed to every recursive call for labeling the children $\mu_1, \ldots, \mu_m$ of μ. This ensures that at every descendant of μ the skeleton is considered to be flipped. The order in which the function recurses and labels is illustrated by the arrow in the sketch to the right of Algorithm 2. As in the proof of Theorem 3.10, we assume that s and t are the poles of the current tree node μ. The reference edge (s, t) is drawn as an arc from s to t. The global counter is represented by the global variable *cnt* that is used to assign the final label $\pi(v)$ to a vertex v.

Similar in the P-node case that is described in more detail in Algorithm 3, the value of *isFlipped* is being propagated along the subtree. However, the P-node case requires an additional change, where a possible Q-node must be moved

```
procedure labelPNode(μ, isFlipped)
begin
    // Ensure that a possible Q-node is μ_1
    μ_j ← μ_i with 2 ≤ i ≤ m and μ_i is a Q-node;
    if μ_j ≠ ∅ then
        swapEdges(e_i, e_j)
    end
    for i ← m down to 1 do
        labelMain(μ_i, isFlipped);
    end
end
```



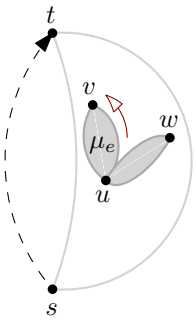
Algorithm 3: Labeling in the P-node case.

such that it follows the reference edge. Therefore, we check all children except the first for being such a Q-node. The function swapEdges then changes the embedding such that the first child μ_1 is a possible Q-node. Afterwards, the function recurses in reverse order on the children.

The R-node case, the one that may initiate a flip along the poles $\{s, t\}$, is the most involved one. Here we first compute a canonical ordering and take its vertex ordering (see the proof of Lemma 3.8) as a temporary ordering π'. Then we traverse the vertices of the skeleton in that order and, as described in the proof of Theorem 3.10, at a vertex v, we first recurse on those children μ_e that are represented by incoming edges with respect to π', that is, all $e = (u, v)$ with $\pi'(u) < \pi'(v)$, before labeling v itself. Recall that in the proof we distinguished two cases depending on whether v is in the increasing partition in the successor list $S'(u)$ of its predecessor u or not. In the first case the embedding of the corresponding children must be flipped. To test for this condition, it is sufficient to check, if the clockwise neighbor has a higher label. Furthermore, one has to keep in mind that when considering one particular child whose subtree embedding must be flipped, the skeleton of μ itself may have already been marked to be flipped. Hence, no action must be taken for that child. Therefore, it is sufficient to just negate the value of *isFlipped*.

Implementation details The algorithm has been implemented in C++ using the Open Graph Drawing Framework (OGDF) [65]. For the canonical ordering required in the R-node case, we implemented the *leftist canonical ordering* algorithm as described by Badent et al. [4]. The linear-time implementation of Gutwenger and Mutzel [52] is used for the SPQR-tree that provides, besides a convenient way to traverse the skeletons and tree, the functionality to propagate the changes made to the embedding.

```
procedure LABELRNODE(μ, isFlipped)
begin
    π' ← vertex ordering from canonical ordering of G_μ;
    for i ← 1 to |V_μ| do
        v ∈ V_μ with π'(v) = i;
        for e = (u, v) ∈ E_μ with π'(u) < π'(v) do
            w ← clockwise neighbour of v at u;
            // Determine if we require an increasing or
               decreasing sequence.
            if π'(w) > π'(v) then
                LABELMAIN(μ_e, not isFlipped);
            else
                LABELMAIN(μ_e, isFlipped);
            end
            if v ≠ t then
                π(v) ← cnt;
                cnt ← cnt + 1;
            end
        end
    end
end
```



Algorithm 4: Labeling the triconnected skeleton in the R-node case.

3.1.2 The fixed embedding scenario

In a preliminary version of this work [50], we presented a simple counterexample that, for a fixed embedding, and fixed *st*-edge does not admit a bitonic *st*-ordering. However, when choosing a different edge as the *st*-edge, a feasible ordering can be found. In the following we prove a slightly stronger result by allowing a free choice of s and t. We even lift the restriction that they are incident to the same face.

Furthermore, one may get the impression that a lower maximum degree may simplify things. For example, in a 3-planar graph, every internal node $v \in V$ with $v \neq s$ has at least one predecessor and as a result at most two successors. Since it is not possible to violate the bitonic property with two elements, because two different elements are either sorted increasingly or decreasingly, the bitonic property cannot be violated. Thus, it is tempting to investigate planar graphs with bounded degree like the 4-planar graphs. Unfortunately, it turns out that this is a hopeless endeavor.

But before we construct a counterexample and give a corresponding proof, we introduce the following technical lemma.

▶ **Lemma 3.11.** *Given a biconnected graph $G = (V, E)$ and an st-orientation of G for $s, t \in V$ with $s \neq t$. Let $S \subset V$ be a non-empty subset of the vertices of G with $|\delta(S)| = 2$[1], that is, there are only two edges having one endpoint in S and the other in $V - S$. If $s, t \notin S$ then in the st-orientation one of the two edges $\delta(S) = \{e_1, e_2\}$ is directed towards S and the other towards $V - S$.*

Proof. Assume that both edges, e_1 and e_2, are directed towards S and let $v \in S$ be a vertex for which by construction $v \neq t$ holds. Since in an *st*-orientation there exists at least one path $u \rightsquigarrow t$ for every $u \in V$ with $u \neq t$, the same applies for v. But this path does not exist because the only two edges between S and $V - S$ point towards S. A similar contradiction can be derived for the case in which e_1 and e_2 point towards $V - S$. □

As a next step, we describe a family of graphs for which in the fixed embedding scenario no bitonic *st*-ordering exists. At first glance this family belongs to a very restricted class of planar graphs. On the other hand this class has some properties that answer a few additional questions:

1. Bounded degree graphs: Do 4-planar graphs with a fixed embedding admit a bitonic st-ordering?

2. Connectivity: Lemma 3.8 states that triconnected planar graphs always allow a bitonic st-ordering. What about series-parallel graphs?

3. Choice of s and t: Theorem 3.10 requires s and t to be adjacent. Does a free choice of s and t simplify the problem?

With the following lemma we settle all three questions.

▶ **Lemma 3.12.** *There exists an infinite class of biconnected series-parallel 4-planar graphs that do not admit a bitonic st-ordering for a fixed planar embedding and an arbitrary choice for s and t.*

Proof. The construction is based on the subgraph $G^* = (V^*, E^*)$ and its embedding displayed in Figure 3.6a. We proceed as follows: First, we show that if s and t are not part of G^*, then the induced *st*-orientation is either exactly as displayed in Figure 3.6b or exactly reversed. In both cases, which are symmetric, we show that a bitonic ordering cannot be found. As a last step, we describe a simple family of graphs that contains multiple copies of G^* and at least one of the copies neither contains s nor t.

[1] With $\delta(S)$ we denote the edges that are in the cut induced by $S \subset V$, that is, $\delta(S) = \{(u, v) \in E \mid u \in S \wedge v \notin S\}$.

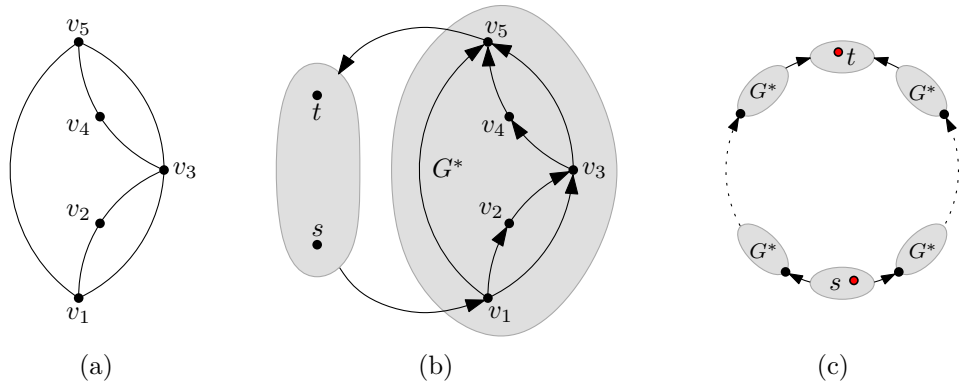


Figure 3.6. (a) Subgraph G^* used in the construction of a counterexample in Lemma 3.12. (b) One of the two possible orientations of G^* when both, s and t are located in another subgraph that is attached by two edges. (c) Construction of G_k with $k \geq 3$.

Let $G = (V, E)$ be an embedded biconnected planar graph that contains G^* as a subgraph, which is attached exactly as in Figure 3.6b, that is, $\delta(V^*) = \{(u_1, v_1,), (u_2, v_5)\}$ with $u_1 \neq u_2$ holds, and the embedding is as shown in Figure 3.6b. Now consider an st-orientation of G for some $s, t \in V - V^*$. For the two edges connecting G^* with $V - V^*$, we may use Lemma 3.11 with $S = V^*$ and assume w.l.o.g. that their orientation is as illustrated in Figure 3.6b. Now it is not difficult to see that every path to t from a vertex $v_i \in V^*$ has to use v_5. Similarly, every path from s to v_i has to contain v_1 and eventually all edges of G^* are oriented towards v_5. More specifically, we may apply Lemma 3.11 two times: First with $S = \{v_4\}$ and then with $S = \{v_2\}$. Clearly, the resulting orientation implies an orientation for the edges (v_3, v_5) and (v_1, v_3). However, if their orientation is (v_5, v_3) and (v_1, v_3), then v_3 is the sink t which by assumption is not the case. Similarly, when (v_3, v_5) and (v_3, v_1) holds, v_3 would be the source s. Now assume that their orientation is (v_5, v_3) and (v_3, v_1). Since an st-orientation is acyclic, we may deduce that $(v_5, v_1) \in E$ which implies that v_1 has only incoming edges, thus would be the sink t, and v_5 has only outgoing edges implying $v_5 = s$ which contradicts our assumption that $s, t \notin V^*$ holds. The only remaining orientation is the one displayed in Figure 3.6b or the reversed one. W.l.o.g., let the orientation be the one in Figure 3.6b. Then in any st-ordering π, $\pi(v_2) < \pi(v_3) < \pi(v_4)$ must hold. And since the successor list of v_1 as implied by the embedding is $S(v_1) = \{v_5, v_2, v_3\}$, we get a sequence $\pi(v_5) > \pi(v_2) < \pi(v_3)$ that is not bitonic increasing. For the reversed orientation, the same arguments can be applied for $S(v_5)$.

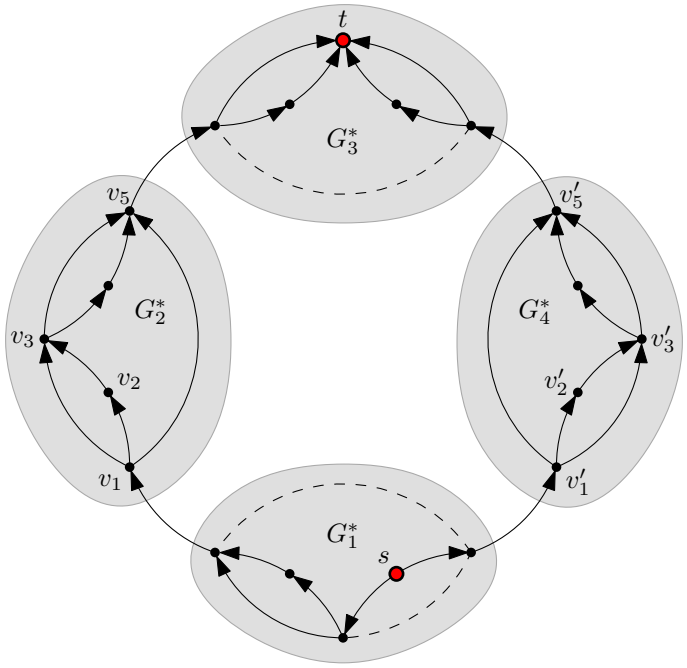


Figure 3.7. Example for G_k with $k = 4$ and its orientation for an arbitrary choice of s and t. Edges without unique orientation are drawn dashed. The successor lists of v_1 and v_1', that is, $S(v_1) = \{v_3, v_2, v_5\}$ and $S(v_1') = \{v_5', v_2', v_3'\}$ are not bitonic (increasing) with respect to any st-ordering.

We may conclude that if neither the source s nor the sink t is located in the subgraph G^*, then only two orientations of the edges in G^* are possible, both not admitting a bitonic st-ordering.

It remains to show that we can construct a family of graphs for which this is always the case. The idea is straightforward: s and t can only occupy at most one copy of G^* each. Hence, we may create a graph G_k that is a a cycle of $k \geq 3$ copies of G^* as depicted in Figure 3.6c. Then for every choice of s, t there are at least $k-2$ successor lists that are not bitonic. A complete example for $k = 4$ is given in Figure 3.7. Clearly, every G_k is 4-planar. Moreover, it is not hard to verify that the graph is series-parallel. □

Before we turn our attention towards possible applications of bitonic st-orderings, we briefly summarize the results obtained so far. With the properties of canonical orderings in mind, we defined a special st-ordering that can be obtained for any biconnected planar graph. Based on the proof of Theorem 3.10, a linear-time algorithm has been described that uses the SPQR-tree approach to

derive an ordering and employs the canonical ordering for the triconnected case. One drawback is that the algorithm is not able to preserve the initial embedding of the graph. However, we have shown that a change in the embedding might be necessary by presenting a family of graphs for which in the fixed embedding scenario no bitonic *st*-ordering exists. Although in the next section we will focus more on putting our new tool to work, we will derive some more interesting properties.

3.2 Straight-line drawings

We start with a classic problem: planar straight-line drawings. We already argued before that the bitonic property is a requirement for many incremental drawing algorithms. Since the algorithm of de Fraysseix, Pach and Pollack [34] is one of them, and many algorithms share the basic concept of creating a drawing from bottom to top in an incremental manner, we show that we can adapt this concept to the bitonic *st*-ordering. Although the ability to draw biconnected planar graphs with a variant of that algorithm is exactly what Harel and Sardas [53] do, thus, the result is not new, we are able to derive a slightly different drawing with the additional property that every face is y-monotone.

The proposed algorithm will later be used to draw directed graphs. Therefore, we describe it here in more detail. Starting with the work of de Fraysseix et al. [34], we review the key concepts of the original algorithm. Afterwards, we describe the variant of Chrobak and Payne [29] that serves as a basis for a linear-time algorithm. Based on their work, an adaptation to bitonic *st*-orderings is presented. At this point it should be mentioned that there exists an extensive amount of literature that is concerned with improvements, variants and extensions of the original algorithm. However, the focus of this chapter lies on bitonic *st*-orderings, therefore we restrict ourselves to the work that is essential for our undertaking.

3.2.1 The algorithm of de Fraysseix, Pach and Pollack

Let us recall the original algorithm of de Fraysseix, Pach and Pollack [34]. Their work marks the beginning of canonical orderings and their use in graph drawing. They introduce the canonical ordering for maximal planar graphs and give a linear-time algorithm to obtain it. This ordering is then used to construct a planar straight-line grid drawing in an incremental manner.

Let $G = (V, E)$ be a maximal planar graph embedded in the plane and $v_1, v_2, \ldots, v_n$ a canonical ordering of the vertices as described in Definition 3.1.

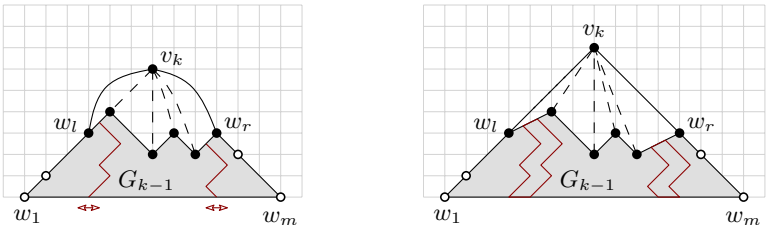


Figure 3.8. (a) Initial situation before installing v_k with neighbors $w_l, \ldots, w_r$. (b) Final layout for G_k after inserting gaps of unit length between w_l, w_{l+1} and w_{r-1}, w_r and placing v_k.

With a slight abuse of notation, we use $C_k = \{w_1 = v_1, \ldots, w_m = v_2\}$ for the path that results from removing the edge (v_1, v_2) from the clockwise cycle that bounds the exterior face of $G_k = (E_k, V_k)$ for $3 \leq k \leq n$. Moreover, we refer to C_k in an informal manner as the *contour* of G_k.

The overall idea of the algorithm can be sketched as follows. The vertices are placed step by step as provided by the ordering. After every step $k > 2$, the layout produced so far must satisfy the following invariant properties:

1. The drawing is a feasible planar straight-line grid drawing for G_k.
2. v_1 is located at the origin $(0, 0)$ and v_2 at $(2k - 4, 0)$.
3. For the vertices of C_k it holds that $x(w_1) < \cdots < x(w_m)$.
4. All edges (w_i, w_{i+1}) with $1 \leq i < m$ have slope $+1$ or -1.

Starting with the first three vertices v_1, v_2, v_3 that are placed at $(0, 0)$, $(2, 0)$, $(1, 1)$, an initial layout for G_3 is created. Clearly, the triangle complies with the invariant.

Now consider the case in which we want to place vertex v_k with $4 \leq k \leq n$. We may assume that the drawing for G_{k-1} satisfies the invariant. The task is to merge v_k into the contour such that the same properties hold also for G_k. By definition, v_k has at least two neighbors on the contour C_{k-1} and they appear consecutively. Let these neighbors be $w_l, \ldots, w_r$ such that $1 \leq l < r \leq m$ holds. Clearly, only the edges (w_l, v_k) and (w_r, v_k) are part of the new contour C_k. Moreover, all vertices between w_l and w_r on C_{k-1} are replaced by v_k. See Figure 3.8a for an illustration.

The problem arising at this point is to find a suitable position for v_k such that all $w_l, \ldots, w_r$ are visible from v_k, and at the same time the edges (w_l, v_k)

and (w_r, v_k) have the required slopes. A straightforward solution is to install v_k above $w_l, \ldots, w_r$ such that (w_l, v_k) has slope $+1$, whereas (w_r, v_k) has slope -1. The corresponding coordinates can be obtained by

$$\begin{aligned} x(v_k) &= \frac{x(w_r) + x(w_l) + y(w_r) - y(w_l)}{2}, \\ y(v_k) &= \frac{x(w_r) - x(w_l) + y(w_r) + y(w_l)}{2} \end{aligned} \tag{3.1}$$

that can be proven to be a grid point due to the invariant properties. Of course, this only works if w_r and w_l are visible from v_k. This, however, is not always the case. By the invariant, the edge (w_l, w_{l+1}) may have a slope of $+1$, but then it overlaps (w_l, v_k). See Figure 3.8a for an example. Similarly, if the slope of (w_{r-1}, w_r) is -1, then it overlaps (w_r, v_k). Notice that the remaining neighbors w_i with $l < i < r$ are all visible from v_k.

The solution to this problem is to modify the drawing of G_{k-1} before installing v_k such that the contour is being stretched in two places, namely directly to the right of w_l and directly to the left of w_r. More specifically, we insert a horizontal gap of unit length between w_l and w_{l+1}, and w_{r-1} and w_r. Then v_k is placed, and as a result w_l is visible from v_k, because the edge (w_l, w_{l+1}) has a slope of less than one now, that is, (w_l, w_{l+1}) is below (w_l, v_k). A symmetric argument can be applied for w_r (Figure 3.8b).

Performing such a stretch is not a trivial task, but in [34] an elegant solution for this problem is described. We only outline the approach here. Assume that we have to insert a gap between w_l and w_{l+1}. Clearly, it is sufficient to shift w_{l+1} one unit to the right. To ensure that the resulting drawing still complies with our invariant (except for the edge (w_l, w_{l+1}) which has a different slope now), we may have to shift additional vertices to the right. To do so, we maintain for every w_i on C_k a subset of vertices $M(k, w_i) \subseteq V_k$ containing w_i itself and all vertices that have to be shifted to the right with it. More specifically, these sets satisfy the following properties:

1. $w_j \in M(k, w_i)$ if and only if $j \geq i$
2. $M(k, w_1) \supset M(k, w_2) \supset \cdots \supset M(k, w_m)$

In [34] the authors give a recursive definition of these sets and prove that using them for the shifting procedure results in a feasible drawing. We only give a short description here, a full proof can be found in [34].

Starting with the initial triangle G_3 with contour $C_3 = \{v_1, v_3, v_2\}$, we choose $M(3, v_1) = \{v_1, v_3, v_2\}, M(3, v_3) = \{v_3, v_2\}$ and $M(3, v_2) = \{v_2\}$. Now assume, we want to install v_k for $k \geq 4$. By an inductive argument one can assume

that every $M(k-1, w_i)$ with $1 \leq i \leq m$ can be used to perform a shift. Hence, shifting w_{l+1} and w_r by one unit to the right using $M(k-1, w_{l+1})$ and $M(k-1, w_{l+1})$, respectively, enables us to install v_k. And as a result, we get a new contour C_k consisting of the vertices $w_1, \ldots, w_l, v_k, w_r, \ldots, w_m$. It remains to determine their new sets for the shifting procedure which is done in the next step:

$$\begin{array}{lcll} M(k, w_i) & = & M(k-1, w_i) \cup \{v_k\} & \text{for } i \leq l \\ M(k, v_k) & = & M(k-1, w_{l+1}) \cup \{v_k\} & \\ M(k, w_j) & = & M(k-1, w_j) & \text{for } r \leq j \end{array} \tag{3.2}$$

The overall procedure is straightforward now. We described the base case G_3. Let $C_{k-1} = \{w_1, \ldots, w_l, \ldots, w_r, \ldots, w_m\}$ be the contour after step $k-1$ with $w_l, \ldots, w_r$ being the neighbors of v_k on C_{k-1}.

1. For each $v \in M(k-1, w_{l+1})$ do $x(v) := x(v) + 1$.
   For each $v \in M(k-1, w_r)$ do $x(v) := x(v) + 1$.
2. Place v_k (Equation 3.1) and set $C_k = \{w_1, \ldots, w_l, v_k, w_r, \ldots, w_m\}$.
3. For each $v \in C_k$, initialize $M(k, v)$ as described in Equation 3.2.

From there on in every step k two stretches are performed to install v_k. Furthermore, by the definition of the sets it follows that $w_m = v_2$ is shifted two times per step. In total $n-3$ steps are required, resulting in a layout in which v_1 is located at the origin and v_2 at $2n-4$. Moreover, v_n has been installed at $(n-2, n-2)$, thus, being the top most vertex. Clearly, the drawing occupies an area of $(2n-4) \times (n-2)$.

A naive implementation requires quadratic time, but de Fraysseix et al. [34] present an algorithm that requires only $\mathcal{O}(n \log n)$ time. Instead of describing it here, we turn our attention to the more widely used variant of Chrobak and Payne [29] that runs in linear time and is conceptually simpler.

3.2.2 The linear-time variant of Chrobak and Payne

In their work [29], Chrobak and Payne present an algorithm that uses the basic framework of the original algorithm, but due to their clever management of the information required for the computation of the coordinates, they are able to derive a linear-time implementation.

For a better understanding, we first discuss the obstacles on the road towards a linear-time algorithm, and how Chrobak and Payne overcome these. Clearly, embedding G and computing a canonical ordering can be accomplished in linear time and is not a problem. Furthermore, identifying in every step the neighbors

of v_k and maintaining a contour in a dynamic manner, can be achieved within a reasonable time frame as well. However, the sets required for the shifting procedure pose both, in terms of runtime and space, a problem. Notice that such a set may contain nearly all the vertices of G_{k-1} in the worst case. Besides of the space requirement, a single shift induces a coordinate update for all those vertices, and some of these coordinates are required for the placement of new vertices in subsequent steps.

The key observation made by Chrobak and Payne is that installing a vertex v_k can be accomplished by knowing only the relative horizontal distances between consecutive vertices of C_{k-1} instead of absolute x-coordinates. Let the horizontal distance of w_i to its predecessor w_{i-1} be $\Delta x(w_i) = x(w_i) - x(w_{i-1})$. When placing v_k, we compute $y(v_k)$ as usual, but instead of $x(v_k)$, we calculate $\Delta x(v_k)$. For the latter one we get

$$\Delta x(v_k) = \frac{x(w_r) - x(w_l) + y(w_r) - y(w_l)}{2}.$$

Since we do not know the absolute x-coordinates $x(w_r)$ and $x(w_l)$, we have to express $x(w_r) - x(w_l)$ by relative ones, which can be accomplished by writing

$$\begin{aligned} x(w_r) - x(w_l) &= \sum_{i=l+1}^{r} x(w_i) - x(w_{i-1}) \\ &= \sum_{i=l+1}^{r} \Delta x(w_i). \end{aligned}$$

We define $\Delta x(w_l, w_r) = \sum_{i=l+1}^{r} \Delta x(w_i)$ and obtain

$$\Delta x(v_k) = \frac{\Delta x(w_l, w_r) + y(w_r) - y(w_l)}{2}.$$

In a similar way, for the absolute y-coordinate of v_k we get

$$y(v_k) = \frac{\Delta x(w_l, w_r) + y(w_r) + y(w_l)}{2}.$$

We conclude that $\Delta x(v_k)$ and $y(v_k)$ can both be found without knowing the absolute x-coordinates. Moreover, evaluating $\Delta x(w_l, w_r)$ takes time depending on the number of edges between v_k and G_{k-1}. Hence, the total time for all $4 \leq k \leq n$ is bounded by $|E| \leq 3|V| - 6$. It remains to show that we can benefit from this in terms of efficiency, while at some point being able to derive absolute x-coordinates. Notice that during a shift, the absolute x-coordinates

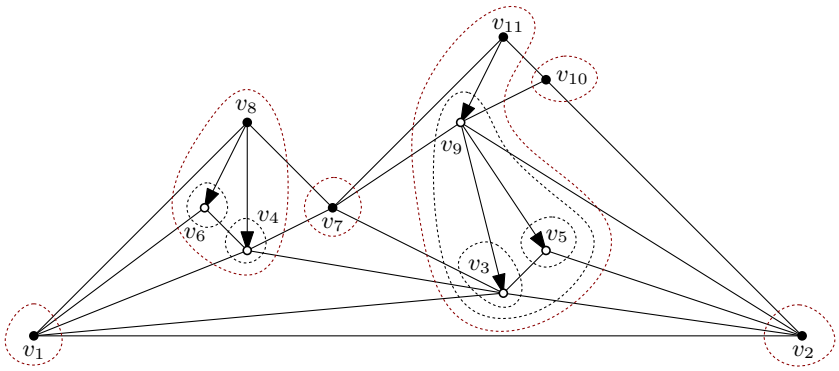


Figure 3.9. Example for the nested hierarchy of shifting sets (dotted). The implied tree is indicated by the arcs that point from the parent to the children.

of every vertex that has to be moved increases, whereas their relative horizontal distances do not change. For example, consider the vertices on the contour C_{k-1}. When installing v_k, we have to shift w_r and all w_i with $r < i \leq m$ with it. While $\Delta x(w_r)$ increases, every $\Delta x(w_i)$ with $r < i \leq m$ remains unchanged. Hence, the vertices of the contour can be shifted efficiently, because the placement of v_k induces only updates of $\Delta x(w_{l+1})$ and $\Delta x(w_r)$. However, this works only for the vertices that are part of C_{k-1}.

For shifting the vertices that are located below the contour, Chrobak and Payne introduce *shifting sets* similar to those used by de Fraysseix et al. [34]. Roughly speaking, a shifting set $L(w_i)$ of a vertex w_i contains the vertices below it, whereas the set $M(k, w_i)$ used in the original algorithm contains those that are below and to the right of w_i. As in [34], these sets are maintained during the incremental drawing procedure. For the first three vertices $v_1, \ldots, v_3$, we set $L(v_i) = \{v_i\}$ for $i \in \{1, 2, 3\}$, and for v_k with $4 \leq k \leq n$, $L(v_k)$ is initialized with

$$L(v_k) = \{v_k\} \cup L(w_{l+1}) \cup \cdots \cup L(w_{r-1}), \tag{3.3}$$

that is, $L(v_k)$ contains v_k itself and the shifting sets of all the vertices that v_k replaces on C_{k-1}. Recall that in the original approach a vertex w_i has different sets depending on k, while here $L(v_k)$ is created in step k and never touched again in any subsequent step. A closer look at the construction of $L(v_k)$ reveals a nested structure of shifting sets (Figure 3.9). In fact one may consider this nested structure as a tree rooted at v_k with $w_{l+1}, \ldots, w_{r-1}$ being the children of v_k. Moreover, every w_i with $l < i < r$ may itself be a root of a subtree. In their work, Chrobak and Payne [29] suggest to encode this tree as a binary tree, but we stick here to an implementation that employs the original tree.

This hierarchy can be exploited for the computation of absolute x-coordinates by making the following observation: After installing the vertex v_k, its children $w_{l+1}, \ldots, w_{r-1}$ are shifted if and only if v_k is shifted. And by an recursive argument this does not hold only for $w_{l+1}, \ldots, w_{r-1}$, but also for every vertex in their shifting sets. Therefore, once v_k is installed, all vertices in $L(v_k)$ keep their relative positions to each other. Let the distance between v_k and one of its children w_i with $l < i < r$ be $\Delta x(v_k, w_i) = x(w_i) - x(v_k)$. Notice that $\Delta x(v_k, w_i)$ can be expressed using relative distances by writing

$$\begin{aligned}\Delta x(v_k, w_i) &= x(w_i) - x(w_l) - x(v_k) + x(w_l) \\ &= \left(\sum_{j=l+1}^{i} \Delta x(w_j) \right) - \Delta x(v_k).\end{aligned}$$

Assume now that we are able to determine $x(v_k)$ in some way. Clearly, knowing $x(v_k)$ and $\Delta x(v_k, w_i)$ is sufficient to determine the absolute coordinate $x(w_i)$. In other words, the absolute x-coordinate of the parent yields the absolute x-coordinates for the children. Hence, when installing v_k, we compute $\Delta x(v_k, w_i)$ for every w_i with $l < i < r$ and set v_k to be the parent of all w_i.

Having placed all vertices, we calculate the absolute x-coordinates for the vertices on $C_n = \{v_1, v_n, v_2\}$ by accumulating their relative coordinates. Then we visit the remaining ones in reverse order as they appear in the canonical ordering, which corresponds to a top-down traversal of the hierarchy induced by the shifting sets. When visiting a vertex, say v, we may assume that the absolute x-coordinate of its parent, say p, has already been obtained and with $x(v) = x(p) + \Delta x(p, v)$ we are able to derive $x(v)$. This way we may obtain absolute coordinates for all vertices.

The complete procedure is given in Algorithm 5. We assume that a maximal planar graph $G = (V, E)$ and a corresponding canonical ordering $v_1, \ldots, v_n$ is given. For the coordinates we use two arrays: While $y(v)$ contains the absolute y-coordinate of v, $x(v)$ is used to store multiple values. As long as v is part of the contour, $x(v)$ contains the relative distance to its predecessor $\Delta x(v)$ on the contour. When v disappears from the contour, we use $x(v)$ to store the horizontal offset to its parent in the shifting set tree. At the end, that is after the top-down traversal, $x(v)$ contains the absolute coordinate of v. It is not hard to see from Algorithm 5 that the procedure takes linear time. Let us summarize the algorithm by stating the following result without a proof.

▶ **Theorem 3.13.** *Every maximal planar graph $G = (V, E)$ admits a planar straight-line drawing of size $(2|V| - 4) \times (|V| - 2)$ which can be obtained using linear time and space.*

```
procedure PlanarStraightLine
begin
    C_3 ← {v_1, v_3, v_2};
    x(v_1) ← 0; y(v_1) ← 0;
    x(v_3) ← 1; y(v_3) ← 1;
    x(v_2) ← 2; y(v_2) ← 0;
    // bottom-up pass
    for k = 4 to n do
        (l, r) ← getLeftRight(v_k, C_{k−1});
        // distance w_l ↔ w_r after shift
        d ← 2 + Σ_{i=l+1}^{r} x(w_i);
        // place v_k
        x(v_k) ← (d + y(w_r) − y(w_l))/2;
        y(v_k) ← (d + y(w_r) + y(w_l))/2;
        // offset w_{l+1}, ..., w_{r−1} ↔ v_k
        t ← 1 − x(v_k);
        for i = l + 1 to r − 1 do
            parent(w_i) ← v_k;
            t ← t + x(w_i);
            x(w_i) ← t;
        end
        // distance v_k ↔ w_r
        x(w_r) ← d − x(v_k);
        C_k ← replace w_{l+1}, ..., w_{r−1} in C_{k−1} with v_k
    end
    // absolute coordinates for the outer face
    for i = 2 to |C_n| do
        x(w_i) ← x(w_i) + x(w_{i−1})
    end
    // top-down pass
    for k = n down to 3 do
        if parent(v_k) then x(v_k) = x(v_k) + x(parent(v_k));
    end
end
```

Algorithm 5: Linear-time implementation of the shifting method. getLeftRight determines the leftmost and rightmost neighbor of v_k on the contour C_{k-1}.

Before we present the adaptation to bitonic *st*-orderings, we make a short detour that is concerned with the computation of coordinates.

A note on coordinate dependencies

The idea of Chrobak and Payne to reduce the dependency of the y-coordinates from absolute x-coordinates to relative ones is essential for their algorithm. However, this does not imply that the y-coordinates depend on any form of x-coordinates at all. Although the slopes required by the invariant suggest that in some way x-coordinates are involved, with a bit of work one can derive an expression that is solely based on y-coordinates. This quite counterintuitive result is not required anywhere in this work. Nevertheless, it is an interesting property by itself that deserves some attention.

Let us assume that for some reason we are interested only in the y-coordinates of the resulting layout. Recall that when installing v_k, we choose

$$y(v_k) = \frac{\Delta x(w_l, w_r) + y(w_r) + y(w_l)}{2}$$

with $\Delta x(w_l, w_r) = \sum_{i=l+1}^{r} \Delta x(w_i)$ being the horizontal distance between w_l and w_r after adjusting the offsets of w_{l+1} and w_r. Now let $\Delta x'(w_i)$ be the initial offset of w_i after step $k-1$, before any adjustments are made to place v_k. Similarly, let $\Delta x'(w_l, w_r) = \sum_{i=l+1}^{r} \Delta x'(w_i)$ be the corresponding distance between w_l and w_r. Since for the placement of v_k two gaps are inserted, both of length one, $\Delta x(w_l, w_r) = \Delta x'(w_l, w_r) + 2$ holds. Hence, we may write

$$y(v_k) = 1 + \frac{\Delta x'(w_l, w_r) + y(w_r) + y(w_l)}{2}.$$

Notice that $w_l, \ldots, w_r$ are consecutive vertices on the contour created in step $k-1$. By the invariant, the slope of every edge (w_{i-1}, w_i) with $1 \leq i \leq m$ is either $+1$ or -1. In the former case, it follows then that $\Delta x'(w_i) = y(w_i) - y(w_{i-1})$, whereas in the latter case, $\Delta x'(w_i) = y(w_{i-1}) - y(w_i)$ holds, that is, $\Delta x'(w_i) = |y(w_i) - y(w_{i-1})|$. Now it is not difficult to see that $y(v_k)$ can be solely derived using y-coordinates:

$$y(v_k) = 1 + \frac{1}{2}\left(y(w_r) + y(w_l) + \sum_{i=l+1}^{r} |y(w_i) - y(w_{i-1})|\right) \tag{3.4}$$

One may further transform this equation using the following observation: Similar to the fact that $\Delta x'(w_l, w_r) = \sum_{i=l+1}^{r} \Delta x'(w_i)$ holds, the vertical difference

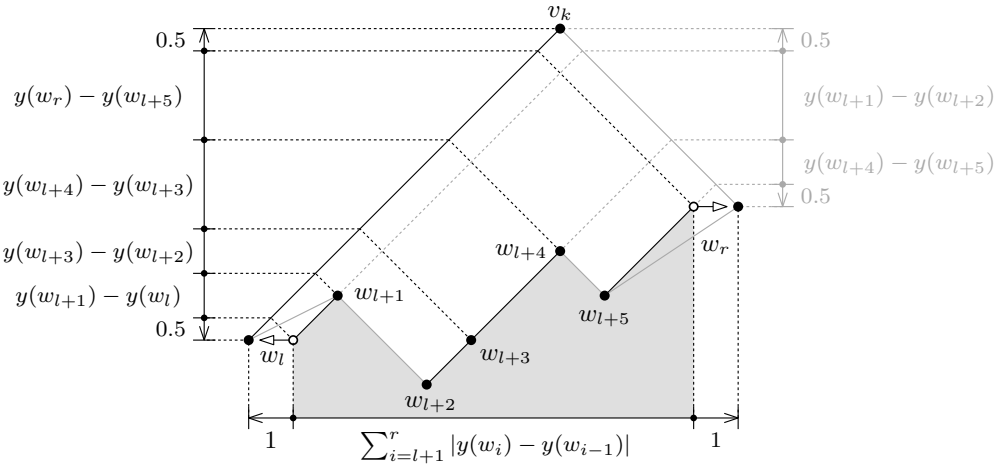


Figure 3.10. Geometric interpretation of the two equations derived for $y(v_k)$. Dotted lines illustrate how the vertical difference $y(w_i) - y(w_{i-1})$ contributes to $y(v_k)$, which depends on if the slope of (w_{i-1}, w_i) is either $+1$ (left, black) or -1 (right, gray). The initial positions of w_l and w_r before the insertion of gaps, are drawn as white circles, whereas the positions after installing v_k are indicated by black circles.

between w_l and w_r can be expressed by

$$y(w_r) - y(w_l) = \sum_{i=l+1}^{r} y(w_i) - y(w_{i-1}).$$

Using this expression to replace $y(w_r)$ in Equation 3.4 yields

$$y(v_k) = 1 + \frac{1}{2}\left(2y(w_l) + \sum_{i=l+1}^{r} (y(w_i) - y(w_{i-1})) + \sum_{i=l+1}^{r} |y(w_i) - y(w_{i-1})|\right),$$

which further simplifies to

$$y(v_k) = y(w_l) + 1 + \frac{1}{2}\left(\sum_{i=l+1}^{r} (y(w_i) - y(w_{i-1})) + |y(w_i) - y(w_{i-1})|\right).$$

With $a + |a| = \max(0, 2a)$ for every $a \in \mathbb{R}$, we get

$$y(v_k) = y(w_l) + 1 + \sum_{i=l+1}^{r} \max(0, y(w_i) - y(w_{i-1})).$$

So instead of accumulating the horizontal offsets, it is sufficient to accumulate the vertical distances of edges with slope +1. Replacing $y(w_r)$ in Equation 3.4 was an arbitrary decision, substituting $y(w_l)$ instead yields

$$y(v_k) = y(w_r) + 1 + \sum_{i=l+1}^{r} \max(0, y(w_{i-1}) - y(w_i)),$$

which accumulates the vertical distance of edges with slope −1 instead of +1. A geometric interpretation of both equations is depicted in Figure 3.10.

Notice that large parts of Algorithm 5 are concerned with the computation of x-coordinates, including the maintenance of the tree and its traversal. Removing the corresponding statements yields a simple algorithm that only computes the y-coordinates (see Algorithm 6).

procedure PLANARSTRAIGHTLINEVERTICAL
begin
  $C \leftarrow \{v_1, v_3, v_2\}$;
  $y(v_1) \leftarrow 0$; $y(v_2) \leftarrow 1$; $y(v_3) \leftarrow 0$;
  for $k = 4$ **to** n **do**
    $(l, r) \leftarrow$ GETLEFTRIGHT(v_k, C);
    $y(v_k) \leftarrow y(w_l) + 1 + \sum_{i=l+1}^{r} \max(0, y(w_i) - y(w_{i-1}))$;
    $C \leftarrow$ replace $w_{l+1}, \ldots, w_{r-1}$ with v_k in C;
  end
end

Algorithm 6: Algorithm for computing only the y-coordinates.

But let us now return to our original undertaking and focus on the development of a straight-line drawing algorithm for bitonic *st*-orderings.

3.2.3 A drawing algorithm for bitonic *st*-orderings

In the following, we describe how to adapt the algorithm from the previous section to bitonic *st*-orderings by borrowing some ideas from Harel and Sardas [53]. They first describe a linear-time algorithm to compute a biconnected canonical ordering as defined in Definition 3.4. Then a modification of the algorithm of de Fraysseix, Pach and Pollack [33, 34] is used to obtain a planar straight-line layout. The key observation is that when installing a vertex v_k that has at least two neighbors on the contour C_{k-1}, one can proceed as in the original algorithm we presented earlier (Algorithm 5).

The only problematic case is the one in which a vertex v_k has only one neighbor on C_{k-1}, say w_i. Recall that Harel and Sardas [53] introduced the property

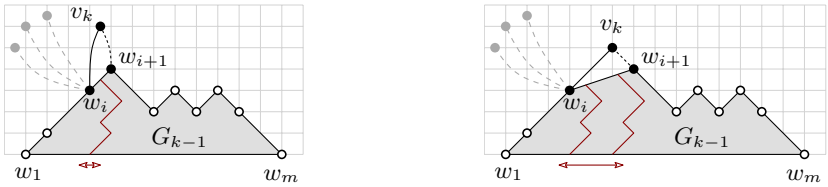


Figure 3.11. (a) A vertex v_k with only one predecessor w_i has right support. Vertices in grey have not been drawn yet. (b) The result of using w_{i+1} as a second neighbor on C_{k-1} in the layout algorithm by setting $w_l = w_i$ and $w_r = w_{i+1}$.

of having *left, right* and *legal support* for these vertices (see Definition 3.3). Their solution to the problem is as follows: If v_k has left support at its only neighbor w_i, then one may use w_{i-1}, the predecessor of w_i on C_{k-1}, as a second neighbor for v_k and proceed as in the original algorithm by pretending that the edge (v_k, w_{i-1}) exists. However, this is only possible, because the property of having left support guarantees that all edges that have to be attached to w_i later, follow (v_k, w_i) clockwise in the embedding. Roughly speaking, all edges to be attached later appear to the right of v_k, so v_k is placed to the left of w_i to keep w_i accessible from above. Similarly, when v_k has right support, every edge incident to w_i that is not yet present will be attached from the left. Therefore, in case of right support, we may use w_{i+1} as a second neighbor for v_k. An example for having right support is given in Figure 3.11.

We already argued that the bitonic *st*-ordering has similar properties. With the following lemma, we gain the discussed property of having left and right support for bitonic *st*-orderings.

▶ **Lemma 3.14.** *Let $G = (V, E)$ be a connected planar graph embedded in the plane with a corresponding bitonic st-ordering π. Moreover, let v_k be the k-th vertex in π and G_k the subgraph induced by $v_1, \ldots, v_k$. For every $1 < k \leq |V|$ the following holds:*

1. *G_k and $G - G_k$ are connected,*
2. *v_k is in the outer face of G_{k-1},*
3. *For every vertex $v \in V_k$, the neighbors of v that are not in G_k appear consecutively in the embedding around v.*

Proof. The first and second statement hold for every *st*-ordering with s and t on the outer face. For the third statement assume to the contrary, that for some $1 < k \leq |V|$ the neighbors of a vertex v with $\pi(v) \leq k$ that are in $G - G_k$ do not

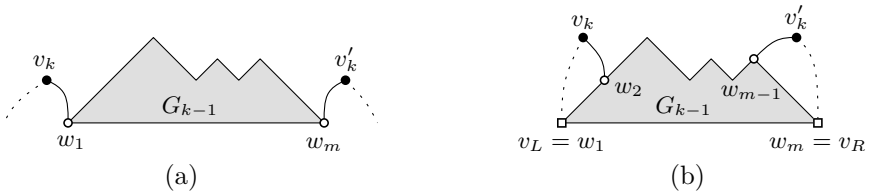


Figure 3.12. (a) The problem of having no legal support at the boundary of the contour $C_{k-1} = \{w_1, \ldots, w_m\}$. The vertex to place has left support at w_1 or right support at w_m. (b) Two artificial vertices v_L, v_R, one at the beginning and one at the end of $C_{k-1} = \{v_L = w_1, \ldots, w_m = v_R\}$ may serve as a second neighbor of v_k in G_{k-1}.

appear consecutively in the embedding around v. Then v has two successors $w_a, w_c \in S(v)$ with $\pi(w_a) > k$ and $\pi(w_c) > k$. Assume that w_a precedes w_c in $S(v)$, that is $a < c$. Since all vertices in $S(v)$ appear consecutively in the embedding, there exists then a third successor w_b between w_a and w_c in $S(v)$ that by our assumption is in G_k, that is, $\pi(w_b) \leq k$ holds. Notice that $S(v)$ is of the form $S(v) = \{\ldots, w_a, \ldots, w_b, \ldots, w_c, \ldots\}$ and $\pi(w_a) > \pi(w_b) < \pi(w_c)$ holds, which contradicts that $S(v)$ is bitonic with respect to π. □

It is not difficult to see that due to the third statement, we can use the idea of Harel and Sardas to deal with the case in which a vertex has only a single predecessor. When placing such a vertex, say v_k, whose only predecessor is u, then we can assume that v_k is not preceded and followed in $S(u)$ by vertices with a label greater than k. Therefore, the concept of having left and right support translates to bitonic *st*-orderings in the following sense: v_k has left support (at u) if no vertex preceding v_k in $S(u)$ exists with a label greater than k. And in a symmetric manner, v_k has right support, if there is no vertex following v_k in $S(u)$ with a label greater than k.

However, one problem arises: The approach by Harel and Sardas requires a vertex with only one neighbor on C_{k-1} to have legal support, not just left or right support. A quick look at Definition 3.3 reveals that there is only a difference at the boundary of the contour. More specifically, if the only predecessor of v_k is w_1 (or w_m), then v_k must have right support (or left support, respectively). This is, however, not necessarily the case in a bitonic *st*-ordering, where it may happen for example that v_k has right support at w_m. Let us assume for a moment that we have to cope with this case in which v_k has right support at w_m. Hence, the edge (v_k, w_m) must have a slope of $+1$, thus, we are forced to choose $w_l = w_m$, whereas for w_r we are then not able to find an appropriate vertex on C_{k-1}. See Figure 3.12a for an illustration of the problem of lacking legal support.

```
procedure getLeftRightBitonic(v_k, C_{k-1} = {w_1, ..., w_m})
begin
    l ← min{i | (w_i, v_k) ∈ E};
    r ← max{i | (w_i, v_k) ∈ E};
    // one predecessor case
    if l = r then
        v_p ← preceding vertex of v_k in S(w_r);
        if v_p = nil or π(v_p) ≤ k then l ← l − 1;
        v_s ← following vertex of v_k in S(w_r);
        if v_s = nil or π(v_s) ≤ k then r ← r + 1;
    end
    return (l, r)
end
```

Algorithm 7: Computation of w_l and w_r for v_k with the one-predecessor case.

To overcome this problem and without limiting the applicability of our bitonic *st*-ordering, we make a small modification to the algorithm. We add two dummy vertices v_L and v_R that take the roles of v_1 and v_2 in the original algorithm with the property that v_L is always the first, and v_R always the last vertex in every contour, that is, for every $1 \leq k \leq n$, $C_k = \{v_L = w_1, \ldots, w_m = v_R\}$ holds. Notice that v_L and v_R are isolated vertices, thus, there exists no v_k whose only predecessor is v_L or v_R, and that has left or right support. Hence, we are always able to find a second neighbor on C_{k-1} for v_k as depicted in Figure 3.12b.

Now we put these ideas together by describing how to modify Algorithm 5 from the previous section. We start by placing v_L, v_1 and v_R at $(0,0), (1,1)$ and $(2,0)$, respectively. In every step $2 \leq k \leq n$, we proceed exactly as in Algorithm 5, only the subroutine for determining w_l and w_r has to be adjusted according to the idea of Harel and Sardas. The procedure displayed in Algorithm 7 takes care of this additional case in which v_k has only one neighbor on C_{k-1}. However, notice that if v_k has left and right support at w_i, then $w_l = w_{i-1}$ and $w_r = w_{i+1}$ is chosen. A complete example is shown in Figure 3.13, in which the drawing for a small graph with seven vertices is created step by step.

Now it is not hard to see that only minor modifications are necessary to get Algorithm 5 to work with a bitonic *st*-ordering instead of a canonical ordering. Besides the changes for the initial phase of Algorithm 5 such that v_L, v_R, v_1 are placed accordingly, some minor corrections to the range of the loops have to be made. See Algorithm 8 for the detailed changes that are necessary. The missing parts in Algorithm 8 should be replaced with the corresponding parts from Algorithm 5.

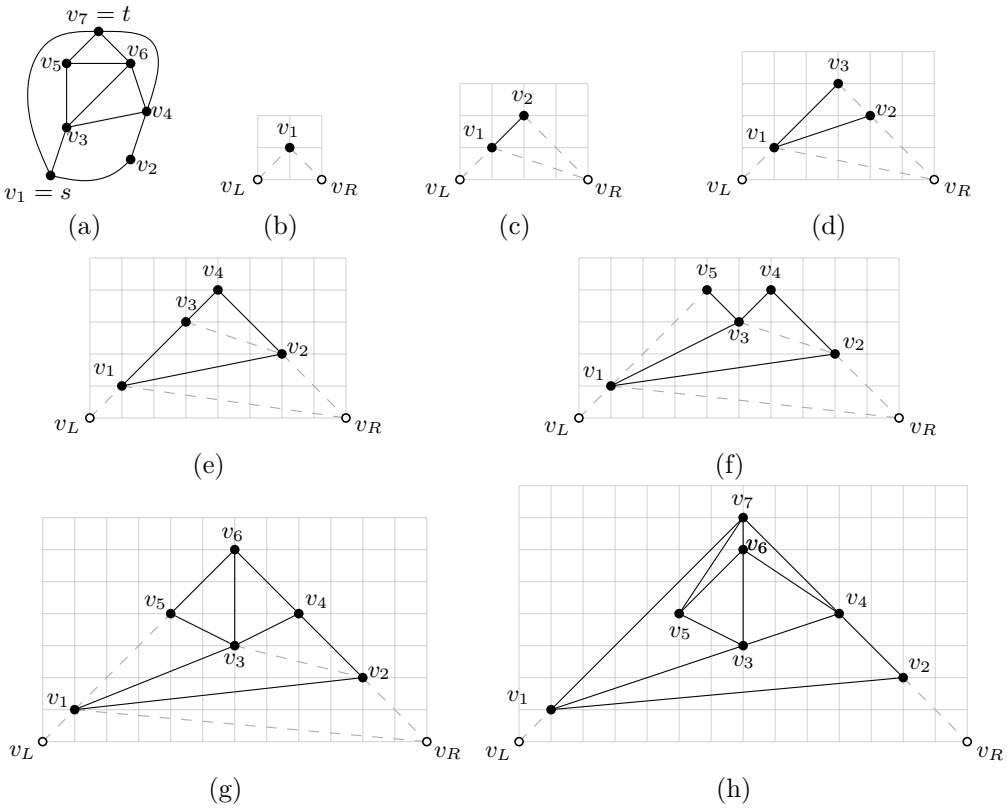


Figure 3.13. (a) Example graph consisting of seven vertices with a bitonic *st*-ordering. (b)-(h) Steps during the construction of the drawing. (c) v_2 is supported by v_R and serves in the next step (d) as supporting vertex for v_3. (f) v_5 uses v_1 as support.

We may conclude that Algorithm 8 enables us to use the idea of de Fraysseix et al., while the algorithm itself is driven by a bitonic *st*-ordering instead of a canonical one. Let us summarize this by stating the following theorem:

▶ **Theorem 3.15.** *Given a plane graph $G = (V, E)$ and a corresponding bitonic st-ordering π for G. A planar straight-line drawing for G of size $(2|V| - 2) \times (|V| - 1)$ can be obtained from π in time $\mathcal{O}(|V|)$.*

Proof. We already argued the runtime, it remains to bound the area. Notice that the input consists besides the vertices of G of two additional vertices v_L, v_R. By Theorem 3.13, the drawing has then a size of $2|V| \times |V|$. However, v_L and v_R are dummy vertices and have to be removed anyway. Moreover, every other vertex is located above them. Hence, their removal yields a slightly smaller drawing of size $(2|V| - 2) \times (|V| - 1)$. □

```
procedure PlanarStraightLineBitonic
begin
    C_1 ← {v_L, v_1, v_R};
    x(v_L) ← 0; y(v_L) ← 0;
    x(v_1) ← 1; y(v_1) ← 1;
    x(v_R) ← 2; y(v_R) ← 0;
    // bottom-up pass
    for k = 2 to n do
        (l, r) ← getLeftRightBitonic(v_k, C_{k−1})
                    ⋮
    // top-down pass
    for k = n down to 1 do ...
end
```

Algorithm 8: Adaptation of Algorithm 5 to bitonic *st*-orderings.

At the beginning of this section we claimed that the resulting drawing has the additional property that every face is y-monotone. For the sake of an easier proof, we will postpone it and show this property as part of the next section.

3.3 Bitonic *st*-orderings of planar *st*-graphs

Unlike in the previous section that was concerned with undirected graphs, we now consider the case in which a directed graph is given. Intuitively this is much more difficult, because every edge is oriented now, generating additional constraints that a bitonic *st*-ordering must comply with. Nevertheless, we start with an important result that motivates the investigation of planar *st*-graphs.

▶ **Theorem 3.16.** *If a planar st-graph admits a bitonic st-ordering, then it admits an upward planar straight-line drawing within quadratic area.*

Proof. Fortunately, the proposed algorithm for drawing undirected graphs in a straight-line style creates such an upward drawing as a byproduct. Notice that the algorithm places every vertex above its predecessors, that is, when placing a vertex $v \in V$, for every $(u, v) \in E$ with $\pi(u) < \pi(v)$, it holds that $y(u) < y(v)$. Since after its placement the y-coordinate is never modified again, it follows that in the final layout $y(u) < y(v)$ holds for every $(u, v) \in E$ with $\pi(u) < \pi(v)$, as required for an upward drawing. Hence, we may apply the algorithm to a planar *st*-graph with a given bitonic *st*-ordering π. By Theorem 3.15, the resulting drawing is a planar straight-line one requiring quadratic area. □

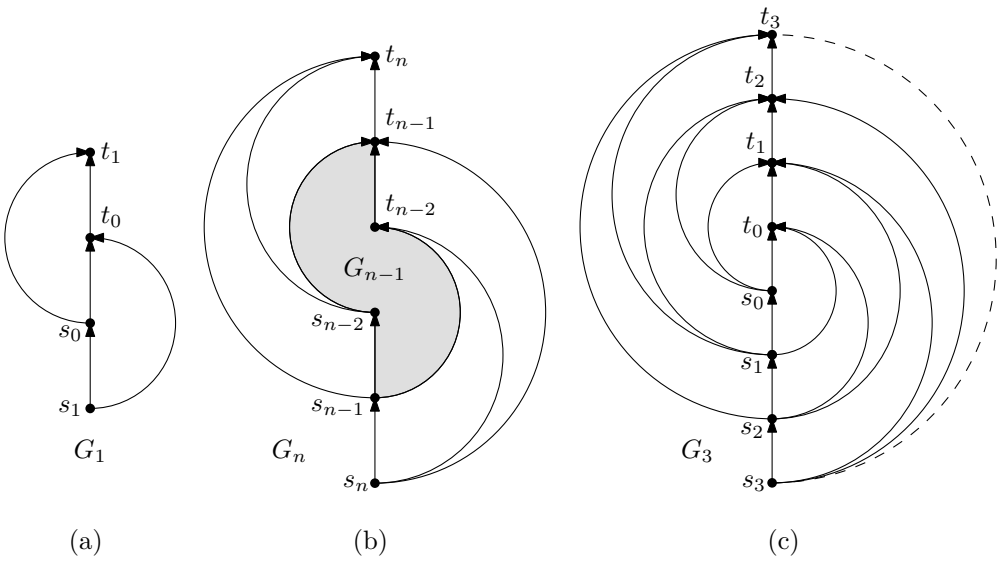


Figure 3.14. (a)-(b) Recursive construction of a planar *st*-graph that requires exponential area in any upward planar straight-line drawing [39]. (c) G_3 in which the successor list $S(s_1)$ and $S(s_2)$ are not bitonic with respect to the only possible *st*-ordering $s_3, \ldots, s_0, t_0, \ldots, t_3$.

Since every face in a planar *st*-graph is bounded by two directed paths, and the drawing is a feasible upward planar drawing, we may argue that every face is y-monotone. Recall that we claimed this in the previous section.

However, to create such a drawing, we must be able to compute a bitonic *st*-ordering for a planar *st*-graph, and the first question that comes to mind is, if this is possible in general or if there exist planar *st*-graphs for which no bitonic *st*-ordering exist. With a closer look at Theorem 3.16, we are able to answer this question immediately. Although every planar *st*-graph admits an upward planar straight-line drawing [37], there exist some classes for which it is known that they require exponential area [36, 39]. The recursive construction [39] of one of these classes and an example is depicted in Figure 3.14. Hence, these graphs cannot admit a bitonic *st*-ordering, because Theorem 3.16 clearly states that the drawing requires only polynomial area. Furthermore, when considering the example G_3 in Figure 3.14c, it is not difficult to see that this particular graph does not admit a bitonic *st*-ordering.

▶ **Corollary 3.17.** *There exist planar st-graphs that do not admit a bitonic st-ordering.*

While this had to be expected, we now have to solve an additional problem.

Before we think about how to compute a bitonic *st*-ordering, we must first be able to recognize planar *st*-graphs that admit such an ordering.

3.3.1 Characterization & recognition

The overall approach can be sketched as follows. We assume a fixed embedding scenario, and show that we can answer our questions without having to think about other embeddings. Based on the bitonic property, we derive a condition that any embedded planar *st*-graph that admits a bitonic *st*-ordering, has to comply with. Afterwards an algorithm is described that exploits this condition to compute a bitonic *st*-ordering, which proves that the condition is sufficient.

We start with an alternative characterization of bitonic increasing sequences, one that suits our needs better in the beginning than the one given earlier in Definition 3.5. Since we will use the labels of an *st*-ordering, we can assume that the elements are pairwise distinct.

▶ **Lemma 3.18.** *An ordered sequence $A = \{a_1, \ldots, a_n\}$ of pairwise distinct elements is bitonic increasing if and only if the following holds:*

$$\forall 1 \leq i < j < n \ : \ a_i < a_{i+1} \vee a_j > a_{j+1}.$$

Proof. Recall from Definition 3.5 that A is bitonic increasing if and only if there exists $1 \leq h \leq n$ such that $a_1 < \cdots < a_h > \cdots > a_n$ holds. We first prove "⇒", that is, if A is bitonic increasing, then there exists no pair i, j with $1 \leq i < j < n$ and $a_i > a_{i+1} \wedge a_j < a_{j+1}$. Assume to the contrary that there exists such a pair. Then from $a_i > a_{i+1}$, it follows that $h \geq i$, and $a_j < a_{j+1}$ yields $j < h$, which contradicts $i < j$. For "⇐" we choose, if it exists, $h = \min\{j \mid a_j > a_{j+1}\}$, otherwise we set $h = n$. By our choice of h, $a_i < a_{i+1}$ holds for every $1 \leq i < h$. Moreover, for every $h \leq j < n$, it must hold that $a_j > a_{j+1}$, because otherwise, there exists $1 \leq h < j < n$ with $a_h > a_{h+1} \wedge a_j < a_{j+1}$. □

In general, a planar *st*-graph may have many different *st*-orderings, some of them being bitonic while others are not. To deal with this problem in a more formal manner, we introduce some additional notation. Given an embedded planar *st*-graph $G = (V, E)$ with s and t on the outer face, we refer with $\Pi(G)$ to all feasible *st*-orderings of G. More specifically, $\Pi(G)$ is a finite set of bijections such that

$$\Pi(G) = \{\pi : V \mapsto \{1, \ldots, |V|\} \mid \pi \text{ is an } st\text{-ordering for } G\}.$$

The number of different *st*-orderings of a planar *st*-graph depends highly on its structure. Starting with the graphs that have only one unique *st*-ordering like

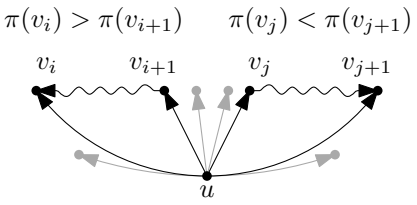


Figure 3.15. A successor list $S(u) = \{\ldots, v_i, v_{i+1}, \ldots, v_j, v_{j+1}, \ldots\}$ with $i < j$ and a forbidden configuration of paths $v_{i+1} \rightsquigarrow v_i$ and $v_j \rightsquigarrow v_{j+1}$.

the planar *st*-graphs that contain a Hamiltonian path (Figure 3.14) to graphs with an exponential number of orderings. For example, the planar *st*-graph with n vertices in which the source and sink are connected by $n-2$ directed paths of length two. Any label between 2 and $n-1$ can be assigned arbitrarily resulting in $(n-2)!$ different *st*-orderings, which represents the upper bound:

$$1 \leq |\Pi(G)| \leq (n-2)!$$

Let $\Pi_b(G)$ be the subset of $\Pi(G)$ that contains all *st*-orderings that are bitonic *st*-orderings. By definition, we can describe $\Pi_b(G)$ by

$$\Pi_b(G) = \{\pi \in \Pi(G) \mid \forall u \in V \ : \ S(u) \text{ is bitonic with respect to } \pi\}.$$

Applying the alternative characterization of bitonicity from Lemma 3.18 to the bitonic property of the successor lists $S(u)$ yields the following expression for the existence of a bitonic *st*-ordering:

$$\begin{aligned} \exists \pi \in \Pi_b(G) \Leftrightarrow \exists \pi \in \Pi(G) \quad & \forall u \in V \text{ with } S(u) = \{v_1, \ldots, v_m\} \\ & \forall\, 1 \leq i < j < m \ : \ \pi(v_i) < \pi(v_{i+1}) \vee \pi(v_j) > \pi(v_{j+1}). \end{aligned} \tag{3.5}$$

Next we translate this expression from *st*-orderings to the existence of paths. Although for now we will only be able to derive a necessary condition for G having a bitonic *st*-ordering, we shall later prove that it is indeed sufficient. Consider a path from some vertex u to some other vertex v in G, then for every $\pi \in \Pi(G)$, by the definition of *st*-orderings, $\pi(u) < \pi(v)$ holds. Now it is not hard to imagine that if there exists $\pi \in \Pi_b(G)$, then there must exist configurations of paths that are forbidden. To clarify this, let us rewrite the last part of the condition in Equation 3.5, that is, $\pi(v_i) < \pi(v_{i+1}) \vee \pi(v_j) > \pi(v_{j+1})$, using a simple boolean transformation, which yields $\neg(\pi(v_i) > \pi(v_{i+1}) \wedge \pi(v_j) < \pi(v_{j+1}))$. So if there exists a path from v_{i+1} to v_i and one from v_j to v_{j+1} with $i < j$, then

this expression evaluates to false for every $\pi \in \Pi(G)$. Therefore, we may refer to the pair of paths $v_{i+1} \rightsquigarrow v_i$ and $v_j \rightsquigarrow v_{j+1}$ with $i < j$ as a **forbidden configuration** of paths. See Figure 3.15 for an illustration.

We may state now that in case there exists a bitonic *st*-ordering, the aforementioned configuration of paths cannot exist:

$$\exists \pi \in \Pi_b(G) \Rightarrow \forall u \in V \text{ with } S(u) = \{v_1, \ldots, v_m\} \\ \forall\, 1 \leq i < j < m \;:\; v_{i+1} \not\rightsquigarrow v_i \vee v_j \not\rightsquigarrow v_{j+1}.$$

Conversely, if we find an u with v_i and v_j in a graph for which these paths exist, then we can safely reject it as one that does not admit a bitonic *st*-ordering. The following well-known property of planar *st*-graphs will prove itself useful when it comes to testing for the existence of a path between two vertices.

▶ **Lemma 3.19.** *Let F be the subgraph of an embedded planar st-graph $G = (V, E)$ induced by a face that is not the outer face*[2]*, and u, v two vertices of F, that is, u and v are on the boundary of the face. Then there exists a path from u to v in G, if and only if there exists such a path in F, that is,*

$$u \rightsquigarrow v \in G \;\Leftrightarrow\; u \rightsquigarrow v \in F.$$

There are several ways to prove this result, one proof can be found in the work of de Fraysseix et al. [32]. They use the term *comparable* stemming from the field of partial order theory, which is equivalent to the existence of a path between u and v. A more graph-based proof is based on the idea that if a path exists that is not part of F, then it must intersect either the path from the face-sink to t or from s to the face-source. Both cases induce a cycle, contradicting *st*-planarity of G.

Notice that Lemma 3.19 is concerned with every pair of vertices incident to the face. But in our case only a special pair of vertices is of interest. Namely, we are interested in the existence of a path between two consecutive successors v_i and v_{i+1} of a vertex u. Notice that v_i, v_{i+1} and u share a common face in which u is the source of that face. Moreover, this face is not the outer face and is located in between v_i and v_{i+1}, that is, v_i is on the left path and v_{i+1} on the right path. Hence, if there exists a path from v_i to v_{i+1}, then v_{i+1} must be the sink of the face (Figure 3.16a). Similarly, if there exists a path from v_{i+1} to v_i, then v_i is the face-sink (Figure 3.16b). Notice that by Lemma 3.19 this holds both ways. Thus, if there exists no path between v_i and v_{i+1}, then none of the two is the sink (Figure 3.16c). As a result, we can test for the existence

[2] This restriction is necessary due to the possible absence of the *st*-edge which is allowed by our definition of planar *st*-graphs.

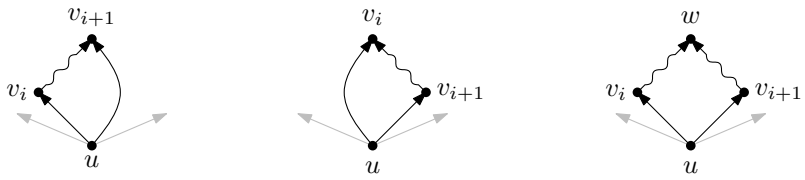


Figure 3.16. The three cases at a face between two successors v_i and v_{i+1} of the face-source u: (a) v_{i+1} is the sink of the face indicating the existence of a path from v_i to v_{i+1}. (b) Similarly, a path from v_{i+1} to v_i results in a face having v_i as sink. (c) There exists no path between v_i and v_{i+1}, if and only if neither v_i nor v_{i+1} is the face-sink.

of a path between two consecutive successors of u by just looking at the sink of their common face with u.

This gives rise to a first idea for an efficient algorithm for our recognition problem. But instead of describing such an algorithm right away, we continue to push for sufficiency. The next proposition is a step towards that in the sense that it essentially reverses the alternative characterization of bitonicity back to the original one. But this time, we do not argue with an *st*-ordering, instead we are able to describe it by the existence of paths.

▶ **Proposition 3.20.** *Given an embedded planar st-graph and a vertex $u \in V$ with successor list $S(u) = \{v_1, \dots, v_m\}$. If it holds that*

$$\forall\, 1 \leq i < j < m \;:\; v_{i+1} \not\rightsquigarrow v_i \vee v_j \not\rightsquigarrow v_{j+1},$$

then there exists $1 \leq h \leq m$ such that

$$(\forall\, 1 \leq i < h : v_{i+1} \not\rightsquigarrow v_i) \wedge (\forall\, h \leq i < m : v_i \not\rightsquigarrow v_{i+1})$$

holds. In other words, there exists at least one v_h in $S(u)$ whose preceding vertices in $S(u)$ are only connected by paths in clockwise direction, whereas paths between following vertices are directed counterclockwise.[3]

Proof. We argue the same way as in the proof of Lemma 3.18. If there exists no path $v_{i+1} \rightsquigarrow v_i$ with $1 \leq i < m$, choose $h = m$. Then $\forall\, 1 \leq i < m : v_{i+1} \not\rightsquigarrow v_i$ is satisfied in a trivial way. If there exists at least one such path, we set $h = \min\{i \mid v_{i+1} \rightsquigarrow v_i\}$ which satisfies $\forall\, 1 \leq i < h : v_{i+1} \not\rightsquigarrow v_i$ by construction.

[3] One can show that this holds both ways, but at this point it is sufficient for our purpose. The other direction will follow implicitly later from the sufficiency of the initial condition.

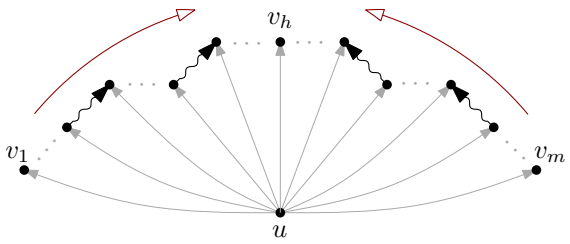


Figure 3.17. Paths and their orientation between consecutive successors of u. All of them directed towards v_h as described by Proposition 3.20.

Now assume to the contrary that there exists a path $v_j \rightsquigarrow v_{j+1}$ with $h \leq j < m$. Then there exists $v_{h+1} \rightsquigarrow v_h$ and $h \leq j$ holds, which contradicts our assumption that for every $1 \leq i < j < m$, it holds that $v_{i+1} \not\rightsquigarrow v_i \vee v_j \not\rightsquigarrow v_{j+1}$. □

The idea is now the following: If we have a graph that satisfies our necessary condition, then we can find for every $u \in V$ with $u \neq t$ a successor v_h with the property as described in Proposition 3.20. The intuition behind this property is that all paths that exist between successors of u, are directed in some way towards v_h. See Figure 3.17 for an illustration. The next lemma exploits this property to obtain a bitonic st-ordering, which proves that this condition is indeed sufficient for the existence of a bitonic st-ordering.

▶ **Lemma 3.21.** *Given a planar st-graph with a fixed embedding. If at every vertex $u \in V$ with successor list $S(u) = \{v_1, \ldots, v_m\}$ the following holds:*

$$\forall\, 1 \leq i < j < m \;:\; v_{i+1} \not\rightsquigarrow v_i \vee v_j \not\rightsquigarrow v_{j+1},$$

then G admits a bitonic st-ordering $\pi \in \Pi_b(G)$.

Proof. To show that there exists a $\pi \in \Pi_b(G)$, we will require several steps. First, we describe an algorithm that augments G into a new planar st-graph by inserting additional edges that we refer to as E'. These edges ensure that between every pair of consecutive successors in G, there exists a path in $G' = (V, E \cup E')$. Afterwards, we prove that G' is st-planar and in the last step, we show that any st-ordering $\pi \in \Pi(G')$ for G' is a bitonic st-ordering for G.

In the following we have to deal with different graphs that share the same vertex set. To avoid any ambiguity of our notation, we establish some rules. The successor list $S(u)$ of a vertex $u \in V$ is used to refer to the successors of u in G as they appear in the embedding of G, and never for any graph other than G. For the existence of a path, we explicitly mention the graph we refer to, for example, $u \rightsquigarrow v \in G$.

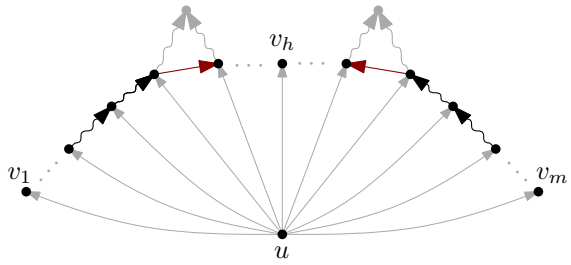


Figure 3.18. The augmented graph G' in the proof of Lemma 3.21 obtained by adding edges between consecutive successors of u such that they are oriented towards v_h.

For every vertex u with successor list $S(u) = \{v_1, \ldots, v_m\}$, we may assume by Proposition 3.20 that there exists $1 \leq h \leq m$ such that for every $1 \leq i < h$ there exists no path from v_{i+1} to v_i, and for every $h \leq i < m$ no path from v_i to v_{i+1} in G. Hence, we may add specific edges to fill the gaps such that there exist two paths in G', $v_1 \rightsquigarrow v_2 \rightsquigarrow \cdots \rightsquigarrow v_h \in G'$ and $v_m \rightsquigarrow v_{m-1} \rightsquigarrow \cdots \rightsquigarrow v_h \in G'$. Figure 3.18 illustrates the idea. More specifically, for every $1 \leq i < m$, there are three cases to consider.

In the first case, there already exists a path between v_i and v_{i+1} in G, that is, $v_i \rightsquigarrow v_{i+1} \in G$ or $v_{i+1} \rightsquigarrow v_i \in G$, hence, we may just skip the pair, because Proposition 3.20 ensures that the path is directed towards v_h. In the second case, there exists no path between v_i and v_{i+1} in G and $i < h$ holds. Having in mind that we want to create a path from $v_1 \rightsquigarrow v_h$ that contains all v_i with $1 < i < h$, we add an edge from v_i to v_{i+1}. In the third case, if there exists no path between v_i and v_{i+1} in G, but now $h \leq i < m$ holds, we add in a symmetric manner the reverse edge (v_{i+1}, v_i) to E'.

As a result of the insertion procedure and the orientation of existing paths as guaranteed by Proposition 3.20, there exists now a path in $G' = (V, E \cup E')$ between every two consecutive successors of u in G such that

$$\forall\, 1 \leq i < h : v_i \rightsquigarrow v_{i+1} \in G' \;\wedge\; \forall\, h \leq i < m : v_{i+1} \rightsquigarrow v_i \in G'.$$

This implies that there exists two paths $v_1 \rightsquigarrow v_2 \rightsquigarrow \cdots \rightsquigarrow v_h \in G'$ and $v_m \rightsquigarrow v_{m-1} \rightsquigarrow \cdots \rightsquigarrow v_h \in G'$.

As a next step, we show that G' is still st-planar by induction over the number of added edges $0 \leq k \leq |E'|$. Assume that the edges in E' are ordered arbitrarily. Let G_k be the graph obtained from inserting the first k edges into G. Clearly, $G_0 = G$ is st-planar.

Now we add the k-th edge. Let this edge be w.l.o.g. (v_i, v_{i+1}), which has been added to E' by visiting two consecutive successors v_i, v_{i+1} of some vertex $u \in V$. The other case in which the k-th edge is (v_{i+1}, v_i) is symmetric. Furthermore, let F be the subgraph induced by the common face of u, v_i and v_{i+1} in G. Notice that F consists of two directed paths, $u \to v_i \rightsquigarrow w$ and $u \to v_{i+1} \rightsquigarrow w$ with face-sink w. By Lemma 3.19, $v_i \neq w \neq v_{i+1}$ holds, because there exists no path between v_i and v_{i+1} in G, which is a requirement for adding an edge. We may assume by our induction hypothesis that G_{k-1} is st-planar. Since we only added edges to G_{k-1}, G is a subgraph of G_{k-1}, and so is F. Moreover, F is still a face in G_{k-1}, because F is the common face of u, v_i, v_{i+1} that we have not touched yet by inserting another edge into F. Hence, we can safely insert (v_i, v_{i+1}) while preserving planarity.

It remains to show that G_k is acyclic. Since G_{k-1} is acyclic, any cycle in G_k must contain (v_i, v_{i+1}). However, then there also exists a path from v_{i+1} to v_i in G_k which must have existed before in G_{k-1}. By Lemma 3.19, there exists then a path in F (and therefore in G) from v_{i+1} to v_i, which is a contradiction. Hence, G_k is st-planar and it follows that G' is st-planar.

Consider now an st-ordering $\pi \in \Pi(G')$. Since clearly $E' \subseteq E \cup E'$ holds, π is also an st-ordering for G, that is, $\Pi(G') \subseteq \Pi(G)$ holds. Recall that we constructed G' such that for every $u \in V$ with $S(u) = \{v_1, \dots, v_m\}$, there exists $v_1 \rightsquigarrow v_2 \rightsquigarrow \cdots \rightsquigarrow v_h \in G'$ and $v_m \rightsquigarrow v_{m-1} \rightsquigarrow \cdots \rightsquigarrow v_h \in G'$. Hence, it follows that for every $\pi \in \Pi(G')$

$$\forall\, 1 \leq i < h : \pi(v_i) < \pi(v_{i+1}) \ \wedge\ \forall\, h \leq i < m : \pi(v_i) > \pi(v_{i+1})$$

holds, which implies that $S(u)$ is bitonic with respect to π. Since this holds for all $u \in V$, it follows that $\Pi(G') \subseteq \Pi_b(G)$. Moreover, G' has at least one st-ordering, that is, $\Pi(G') \neq \emptyset$, thus, there exists $\pi \in \Pi_b(G)$. □

Let us summarize the implications of the lemma. The only requirement is that the graph complies with our necessary condition, that is, the absence of forbidden configurations. If this is the case, then Lemma 3.21 provides us with a bitonic st-ordering, which in turn proves that this condition is sufficient.

$$\exists \pi \in \Pi_b(G) \Leftrightarrow \forall u \in V \text{ with } S(u) = \{v_1, \dots, v_m\}$$
$$\forall\, 1 \leq i < j < m \ :\ v_{i+1} \not\rightarrow v_i \vee v_j \not\rightarrow v_{j+1}$$

Furthermore, the argumentation in the proof of Lemma 3.21 is solely based on Proposition 3.20. This implies that the existence of a bitonic st-ordering, the absence of forbidden configurations and the existence of a v_h in every successor

list as described in Proposition 3.20, are all equivalent.

$$\exists \pi \in \Pi_b(G) \Leftrightarrow \forall u \in V \text{ with } S(u) = \{v_1, \ldots, v_m\} \;\; \exists\, 1 \leq h \leq m \\ (\forall\, 1 \leq i < h : v_{i+1} \not\rightsquigarrow v_i) \wedge (\forall\, h \leq i < m : v_i \not\rightsquigarrow v_{i+1})$$

Additionally, the proof is constructive and yields a linear-time algorithm. Moreover, it is not hard to see that we can easily extend the algorithm to test on the fly, whether the input admits a bitonic *st*-ordering or not. Let us state this as the main result of this section.

3.3.2 Recognition & ordering in linear time

The algorithm is based on two ideas. The recognition process follows the initial idea of testing for forbidden configurations, whereas the ordering uses the idea from the proof of Lemma 3.21. The overall procedure is shown in Algorithm 9 and works as follows. We assume that an embedded planar *st*-graph $G = (V, E)$ is given and wish to test if G admits a bitonic *st*-ordering. In that case the algorithm should compute such an ordering.

At each successor list $S(u) = \{v_1, \ldots, v_m\}$, we test for the existence of the aforementioned special successor v_h. To do so we iterate over $S(u)$ and keep track if the current vertex v_i is in the increasing ($i < h$) or decreasing partition ($h \leq i$). More specifically, if there exists a path $v_i \rightsquigarrow v_{i+1}$, $\pi(v_i) < \pi(v_{i+1})$ holds. So, in case $S(u)$ is bitonic with respect to π, then v_i is in the increasing partition. Once we find a path $v_{i+1} \rightsquigarrow v_i$, we entered the decreasing partition and set the flag accordingly. If we encounter again a path of the form $v_j \rightsquigarrow v_{j+1}$ while the decreasing flag is set, a forbidden configuration has been found and we can safely reject the graph.

During the procedure, we handle the case in which there exists neither a path $v_i \rightsquigarrow v_{i+1}$ nor $v_{i+1} \rightsquigarrow v_i$, the same way as in Lemma 3.21. More specifically, as long as we are in the increasing partition, we add an edge (v_i, v_{i+1}), whereas in the decreasing partition we add the reverse edge (v_{i+1}, v_i) to the set of edges E' to augment G.

In the last step, the bitonic *st*-ordering is obtained by computing an *st*-ordering for the augmented graph $G' = (V, E \cup E')$. Unlike for undirected graphs that require a more sophisticated algorithm, computing an *st*-ordering for a directed graph is a rather simple task. A simple topological sort using a depth-first-search as for example described in [31], is sufficient and takes time $O(|V| + |E| + |E'|)$. Since G' is planar, the total time required is linear in the number of vertices. Let us state this result without a proof.

```
procedure EmbeddedBitonic
begin
    E' ← ∅;
    for u ∈ V with S(u) = {v_1, ..., v_m} do
        decreasing ← false;
        for i = 1 to m − 1 do
            w ← faceSink(u, v_i, v_{i+1});
            if w = v_{i+1} and decreasing then reject;
            if w = v_i then decreasing ← true;
            if v_i ≠ w ≠ v_{i+1} then
                if decreasing then
                    E' ← E' ∪ (v_{i+1}, v_i)
                else
                    E' ← E' ∪ (v_i, v_{i+1})
                end
            end
        end
    end
    compute π ∈ Π(V, E ∪ E');
    return π
end
```

Algorithm 9: Recognition of planar *st*-graphs that admit a bitonic *st*-ordering and its computation. The algorithm assumes that $G = (V, E)$ is an embedded planar *st*-graph and its successor lists are given.

▶ **Theorem 3.22.** *Deciding whether an embedded planar st-graph G admits a bitonic st-ordering π or not is linear-time solvable. Moreover, if G admits such an ordering, π can be found in linear time.*

3.3.3 Experimental results

With a recognition algorithm now at our disposal, we investigate in the following the bitonicity of random planar *st*-graphs. In order to gain some insight into how applicable the concept of bitonic *st*-ordering is, we generated a benchmark set consisting of random planar *st*-graphs of varying size and density. Although we follow the basic principle for generating instances as in the literature, e.g. [11, 25], we give a more detailed description here due to some additional requirements.

Namely, we are interested in generating an embedded planar *st*-graph G that has exactly n vertices and m edges. Of course, we assume that n and m are feasible values. Since in the previous section, G has usually been considered to be simple, we do not allow parallel edges. An implementation that produces

such a graph turned out to be not as straightforward as one would have hoped, thus, for completeness we describe the procedure here.

Generating random planar *st*-graphs

In order to simplify the algorithm, we add one more constraint that is the presence of the st-edge. Notice that by our definition, the st-edge does not have to be present. The only requirement is that s and t are on the same face. However, for simplicity we generate only instances in which this edge is present.

In the following, an algorithm is described that is based on a generator for biconnected planar graphs and implemented as part of the Open Graph Drawing Framework [65]. However, some minor modifications are made such that the result is a planar st-graph. Such an approach has been used by [11, 25] to generate benchmark instances for studies of upward planarity testing algorithms and related problems.

We start by creating a triangle consisting of three edges $(s,t),(s,v),(v,t)$. From there on, we split edges and faces randomly, until the desired size has been reached. When splitting an edge, both the vertex and edge count increases by one, whereas a face split adds only one edge to the graph. For the input n and m, we may assume that $3 \leq n \leq m \leq 3n-6$ holds, since the underlying graph of the result is at least biconnected (due to the (s,t)-edge) and at most maximal planar. The only way to match the vertex count of n, is by performing exactly $n-3$ edge splits. However, these edge splits generate $n-3$ additional edges. Therefore, the remaining edges must be obtained by $m-n-6$ face splits.

Let $G_k = (V_k, E_k)$ be the planar st-graph after k steps. Furthermore, we maintain the invariant that $|V_k| \leq n$ and $|E_k| \leq m-n+|V_k|$ holds. The bound for $|E_k|$ avoids the situation in which we still have to perform $n-|V_k|$ edge splits to generate enough vertices, while the number of edges remaining is less than $n-|V_k|$. Clearly, for G_0, that is a triangle, the invariant holds.

Consider the case $k > 0$, and assume that we are not done yet, that is, at least one edge or vertex is missing and we have to decide, if we split an edge or a face. Splitting an edge randomly is a rather unproblematic operation, because one may choose at any time any edge, except of (s,t), and the result is still a planar st-graph.

A face split, however, is a different story for which two problems have to be solved: First, a split is not necessarily possible for every face, because only faces that are not triangles can be split. Hence, if G_k is maximal planar, we are forced to split an edge instead. Recall that by our assumption $m \leq 3n-6$ holds, therefore, we can perform such an edge split without exceeding the bound for the number of vertices.

In general, a coin is flipped to decide whether an edge or face split is performed. To deal with the maximal planar case and to prevent splitting too many faces, we choose for the probability of an edge split

$$P_{\text{edge}} = \begin{cases} 1 & \text{if } |E_k| = 3|V_k| - 6 \\ \frac{n-|V_k|}{m-|E_k|} & \text{otherwise.} \end{cases}$$

Notice that by our assumption $1 \leq n - |V_k| \leq m - |E_k|$ holds, which implies $0 \leq P_{\text{edge}} \leq 1$. As a result, in the case in which a face split is chosen, there exists at least one face that is not a triangle. By rolling a dice, we choose one of these faces.

Having chosen a face to split, we have to deal with the second problem: Selecting two non-consecutive vertices of a face as endpoints for the new edge randomly, may result in a pair of parallel edges. Notice that we cannot just remove these edges afterwards due to the requirement that $|E| = m$ must hold. The solution to this problem is to first determine all vertex pairs that are feasible candidates for inserting an edge, that is, vertices incident to the face that are not adjacent. Out of this set we choose randomly one pair for the face split. Although this operation preserves planarity, it may introduce a cycle. Therefore, we have to pay some attention on the direction of the edge. To do so we insert the edge into G_k and test both incident faces for a cycle. If such a cycle exists, we reverse the edge. Notice that we do not have to care about a cycle after a reversal, because it would imply a cycle in G_k, which contradicts that G_k is *st*-planar.

Under every circumstance, the described procedure is able to perform an operation. It remains to show that the result complies with the invariant. It is not hard to see that the result G_{k+1} is *st*-planar. Furthermore, since an edge split raises both the edge and vertex count, $|V_{k+1}| \leq n$ and $|E_{k+1}| \leq m - n + |V_{k+1}|$ holds. In the case of a face split, assume that the second bound is not satisfied. Clearly, $|E_k| = m - n + |V_k|$ holds then, which implies that we flipped the coin with $P_{\text{edge}} = 1$ for the probability of an edge split, a contradiction.

Bitonicity of random planar *st*-graphs

We start with the description of the benchmark set that consists of four different classes based on the density of the instances. The graphs of the first three are classfied by their edge-vertex ratio. More specifically, for every graph $G = (V, E)$ it holds that $|E| = \lfloor \delta \cdot |V| \rfloor$, with $\delta \in \{1.5, 2, 2.5\}$ defining the classes. Graphs of the fourth class are maximal planar, that is $|E| = 3|V| - 6$ holds.

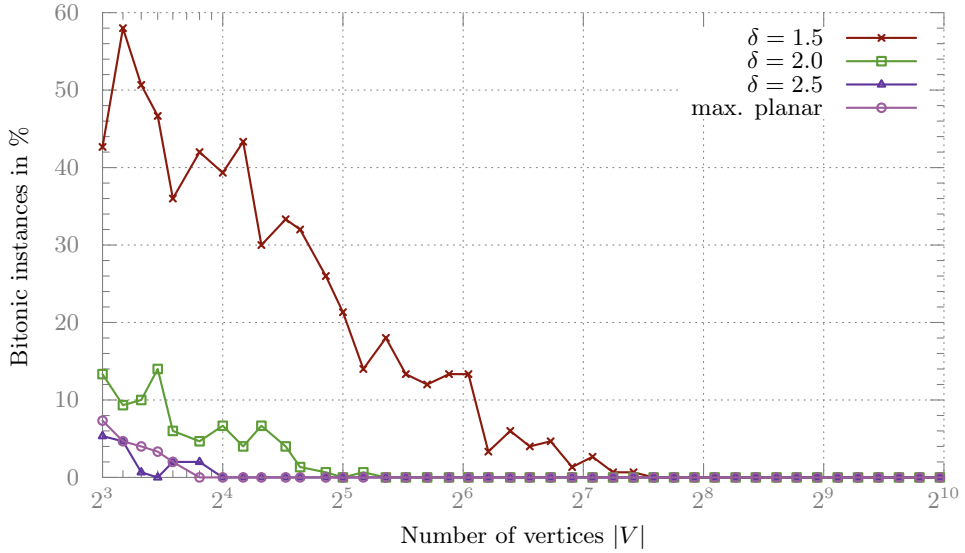


Figure 3.19. Distribution of instances in the benchmark set that admit a bitonic *st*-ordering based on their size.

Every class contains subsets of instances with a vertex count that increases exponentially. More specifically, the k-th subset only contains instances of size $|V| = \left\lfloor \frac{9^k}{8^{k-1}} \right\rfloor$ with $k \in \{0, 1, \ldots\}$. For each such subset we generated 150 sample instances using the generator procedure described earlier. Notice that the described procedure implies an embedding with (s, t) at the outer face.

Figure 3.19 shows the amount of bitonic instances in the corresponding subset, that is the percentage of graphs that admit a bitonic *st*-ordering. The results clearly show what one would expect: With increasing size and density, the amount of bitonic instances decreases rapidly. Notice that the plot has a logarithmic scale for the size. The reason for this decline is that the existence of a single forbidden configuration suffices to reject it. With increasing size and density, it becomes more likely that one of the successor lists contains at least one such configuration.

The distribution of these successor lists is shown in Figure 3.20. To obtain this data, we slightly modified Algorithm 9 such that instead of rejecting an instance right away, it counts the number of successor lists that contain forbidden configurations. The average amount of these lists converges quickly towards a value that solely depends on the edge density.

Regardless of the edge density, the amount of successor lists that are not bitonic shows in the beginning a slightly decreasing trend. This disturbance is

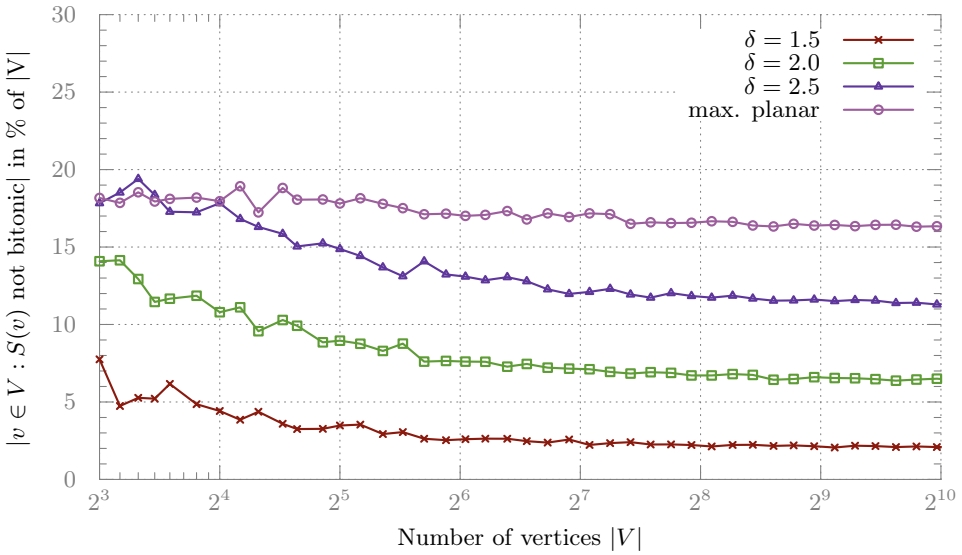


Figure 3.20. Average amount of successor lists that are not bitonic with respect to any *st*-ordering.

caused by the presence of the *st*-edge. Since it is on the outer face, it requires the successor list of s to be sorted instead of being bitonic, which is a slightly stricter requirement. Therefore, the probability that $S(s)$ is not bitonic is higher when the edge (s, t) is present. During our experiments, we temporarily removed the edge from every instance, and we observed a positive effect on the bitonicity of $S(s)$ which supports this claim. Since some instance sets are quite small in terms of their vertex count, a forbidden configuration in $S(s)$ has a higher impact on the total number of successor lists that are not bitonic.

Let us summarize the results so far: Even for rather medium sized graphs ($|V| \approx 256$) with low edge density ($\delta = 1.5$), the probability of admitting a bitonic *st*-ordering is very low. Denser instances ($\delta = 2.5$, maximal planar) with only a few vertices ($|V| \approx 32$) are most likely to contain at least one forbidden configuration. However, the amount of successor lists that contain such configurations is quite low ($< 20\%$). At this point it should be mentioned that our results do not include any runtime data. We omitted these due to the small sizes of the instances and the simplicity of the algorithm.

Now it is tempting to investigate how a variable embedding scenario may improve the number of instances that admit a bitonic *st*-ordering. However, a look at the amount of instances that are maximal planar, thus, have a fixed combinatorial embedding, and do not admit a bitonic *st*-ordering suggests that

this is most likely not the case. Moreover, the amount of successor lists that are not bitonic with respect to any *st*-ordering suggests that the choice of the outer face is not going to help much, because it only affects the successor list $S(s)$. Therefore, the ability to change the embedding cannot have a significant impact on this quite large number of instances.

However, we counteract now in a different way. Recall that our initial motivation was to create upward planar straight-line drawings of planar *st*-graphs. Instead of sticking to straight-line drawings, we allow bends on the edges and shift our efforts to upward planar poly-line drawings.

3.4 Upward planar poly-line drawings with few bends

In general every planar *st*-graph admits an upward planar poly-line drawing in quadratic area. However, the number of bends required varies. An approach that is based on the visibility representation of a planar *st*-graph is described by Di Battista and Tamassia in [37]. Their algorithm produces an upward poly-line drawing by first computing a visibility representation. Based on this visibility representation in which every vertex v is represented by a horizontal segment $x_{\min}(v), x_{\max}(v)$ with y-coordinate $y(v)$ on the grid, they choose for the vertex position an arbitrary grid point that is covered by the corresponding segment, that is, $x_{\min}(v) \leq x(v) \leq x_{\max}(v)$, whereas the y-coordinate $y(v)$ is kept. The edge routing follows a simple principle: every edge $e = (u, v)$ is represented by a vertical segment, say from $x(e), y(u)$ to $x(e), y(v)$ with $y(u) < y(v)$. In the case in which $y(v) - y(u) \geq 1$ holds, this segment serves as a part of a poly-line from u to v by inserting bends at $x(e), y(u) + 1$ and $x(e), y(v) - 1$. Of course, if $y(v) - y(u) = 2$ holds, the result is a poly-line with only one bend, whereas for the case in which $y(v) - y(u) = 1$ holds, no bend is required. Clearly, every edge has at most two bends, therefore, the resulting drawing has at most $6n - 12$ bends with n being the number of vertices of the input graph. With a more careful choice of the vertex positions and by employing a special visibility representation, the authors manage to improve this bound to $(10n - 31)/3$. Moreover, the drawing requires only quadratic area and can be obtained in linear time.

Another approach by Di Battista et al. [39] employs an algorithm that creates a straight-line dominance drawing as an intermediate step. Recall that while a dominance drawing can be transformed into an upward drawing, the converse is not true. We already argued that dominance drawings have much stronger requirements, and that the presented algorithm in [39] cannot handle planar *st*-graphs directly. Instead it requires a reduced planar *st*-graph.

In order to obtain such a graph, Di Battista et al. [39] split every transitive edge of a planar *st*-graph by replacing it with a path of length two. The result is a reduced planar *st*-graph for which a straight-line dominance drawing is obtained that requires only quadratic area and can be computed in linear time. Then they reverse the procedure of splitting the edges by using the coordinates of the inserted dummy vertices as bend points. Since a planar *st*-graph $G = (V, E)$ has at most $2|V| - 5$ transitive edges, the resulting layout has not more than $2|V| - 5$ bends and at most one bend per edge. To our knowledge, this bound is the best achieved so far.

Our situation now is similar: We also have an algorithm for producing an upward planar straight-line layout for a subset of planar *st*-graphs, namely the planar *st*-graphs that admit a bitonic *st*-ordering. And since we put some more effort into an exact characterization of this class, we will be able to derive an algorithm that creates drawings with a small number of bends.

The idea is rather straightforward and can best be described by an example. Assume that an embedded planar *st*-graph $G = (V, E)$ is given that contains a single forbidden configuration of paths at a vertex $u \in V$ with successors list $S(u) = \{v_1, \ldots, v_m\}$. Let this forbidden configuration be two paths $v_{i+1} \rightsquigarrow v_i$ and $v_j \rightsquigarrow v_{j+1}$ with $i < j$ between consecutive successors of u.

Now we augment G by splitting either the edge (u, v_i) or (u, v_{j+1}). Say w.l.o.g. that we split the edge (u, v_i) into two new edges (u, v_i') and (v_i', v_i) with v_i' being the new dummy vertex. Assume that the two new edges inherit their position in the embedding from (u, v_i). As a result, G contains now one additional successor list $S(v_i')$ that contains only one element, thus, is bitonic with respect to every *st*-ordering. Furthermore, v_i has been replaced by v_i' in the successor list $S(u)$, but since there exists no path from v_{i+1} to v_i' anymore, we managed to resolve the forbidden configuration. Now Theorem 3.22 enables us to compute a bitonic *st*-ordering $\pi \in \Pi_b(G)$ in linear time. Running the modified FPP-algorithm (Theorem 3.15) with G and π yields an upward planar straight-line drawing within the same time bound. We reverse the insertion of the dummy vertex by using the position of v_i' as a bend point for the edge (u, v_i), and obtain an upward planar poly-line drawing with exactly one bend for the original graph.

Let us compare this idea with the dominance layout-based approach of Di Battista et al. [39]: In a very similar way, we split an edge to establish a certain property for the graph. Recall that a forbidden configuration consists of two paths $v_{i+1} \rightsquigarrow v_i$ and $v_j \rightsquigarrow v_{j+1}$ with $i < j$ between successors of a vertex u. Then there exists a path $u \rightsquigarrow v_i$ via v_{i+1} that does not contain (u, v_i), and in a symmetric manner, there exists a path $u \rightsquigarrow v_{j+1}$ via v_j that does not contain (u, v_{j+1}). Clearly, (u, v_i) and (u, v_{j+1}) are transitive edges then.

Hence, we may summarize that the existence of a forbidden configuration implies the existence of transitive edges. Conversely, a planar st-graph without these edges cannot contain a forbidden configuration, which immediately yields a subset of planar st-graphs that admit a bitonic st-ordering.

▶ **Corollary 3.23.** *Every reduced planar st-graph admits a bitonic st-ordering.*

Notice that in the example, only one of the two transitive edges has been split to destroy the forbidden configuration of paths, not both of them. Moreover, a pair of transitive edges does not necessarily induce a forbidden configuration. This leads to the question of how many splits are really necessary such that the resulting graph admits a bitonic st-ordering.

3.4.1 The edge-split method

In the following, we give a detailed answer to this question, and as a result, are able to significantly improve the upper bound for the number of bends required in any upward planar poly-line drawing. We proceed in several steps. The first aspect to consider is the case in which we have multiple forbidden configurations and how a split of an edge affects these. Then we prove the upper bound, and as a last step, we describe a linear-time algorithm that splits the minimum number of edges.

Recall that a forbidden configuration of paths is solely based on the existence of paths between the successors of a vertex. The next lemma shows that an edge split has only a very local effect, that is, the existence of paths between any pair of vertices of the original graph is not affected by such a split.

▶ **Lemma 3.24.** *Let $G' = (V', E')$ be the graph obtained from splitting an edge (u, v) of a graph $G = (V, E)$ by inserting a dummy vertex v'. More specifically, let $V' = V \cup \{v'\}$ and $E' = (E - (u, v)) \cup \{(u, v'), (v', v)\}$. Then for all $w, x \in V$ there exists a path $w \rightsquigarrow x \in G$, if and only if there exists a path $w \rightsquigarrow x \in G'$:*

$$\forall w, x \in V : w \rightsquigarrow x \in G \Leftrightarrow w \rightsquigarrow x \in G'.$$

Proof. Notice that $w, x \in V$ implies $w \neq v'$ and $x \neq v'$. Every path in G that contains (u, v) can use $(u, v'), (v', v)$ in G'. Assume there is a path $w \rightsquigarrow x$ in G' that does not exist in G, thus, it contains (u, v') or (v', v). From $w \neq v' \neq x$, it follows that the path contains both edges, (u, v') and (v', v), and that they appear consecutively. Hence, $w \rightsquigarrow x$ can use the edge (u, v) in G instead. □

We can now argue that when splitting an edge, say (u, v), only two successor lists are affected. Namely, $S(u)$ which now contains the newly inserted vertex,

say v', instead of v, and the newly created successor list $S(v')$. However, v' has only one successor, thus $S(v') = \{v\}$ cannot contain a forbidden configuration. Moreover, Lemma 3.24 implies that we are not creating nor resolving any forbidden configurations in other successor lists, because v' is the only new vertex and $S(u)$ is the only successor list that contains it. Hence, no other successor list is affected by this procedure. Since v' has only one predecessor, it is not a face-sink, thus, cannot be involved in other forbidden configurations. Now we exploit this locality by proving an upper bound on the number of edges to split in order to resolve all forbidden configurations.

▶ **Lemma 3.25.** *Every embedded planar st-graph $G = (V, E)$ can be transformed into a new planar st-graph G' that admits a bitonic st-ordering by splitting at most $|V| - 3$ edges.*

Proof. We already argued that the insertion of dummy vertices has only local effects. In order to achieve the upper bound of $|V| - 3$, we follow a simple principle based on the edge-split procedure used in the example to destroy a forbidden configuration.

Consider a vertex u and its successor list $S(u) = \{v_1, \ldots, v_m\}$ that contains multiple forbidden configurations of paths. Intuitively, it is not easy to resolve them within the limits imposed by the upper bound. However, by changing our point of view on the problem, things become more clear. Instead of arguing by means of forbidden configurations, we use our second condition from Proposition 3.20, that is the existence of a vertex v_h such that every path that exists between two consecutive successors v_i and v_{i+1}, is directed from v_i towards v_{i+1} for $i < h$ or from v_{i+1} towards v_i if $i \leq h$ holds. Of course h does not exist due to the forbidden configurations. But we can enforce its existence by splitting some edges.

Assume that we want v_h to be the first successor, that is, $h = 1$. Then every path from v_i to v_{i+1} with $1 \leq i < m$ is in conflict with this choice. However, we can resolve this by splitting every edge (u, v_{i+1}) for which a path $v_i \rightsquigarrow v_{i+1}$ exists. Clearly, the maximum number of edges to split is at most $m - 1$, that is the case in which for every $1 \leq i < m$, there exists a path from v_i to v_{i+1}. However, there do not exist paths $v_i \rightsquigarrow v_{i+1}$ and $v_{i+1} \rightsquigarrow v_i$ at the same time, because G is acyclic. So, if the number of edges to split is more than $\frac{m-1}{2}$, then there are less than $\frac{m-1}{2}$ paths of the form $v_{i+1} \rightsquigarrow v_i$. In that case, we may choose in a symmetric manner h to be the last successor ($h = m$), instead of being the first. Or in other words, we choose h to be the first or the last successor, depending on the direction of the majority of paths. And as a result, at most $\frac{m-1}{2}$ edges have to be split.

Notice that the overall length of all successor lists is exactly the number of edges of the graph. Hence, with $m = |S(u)|$ we get $\sum_{u \in V} |S(u)| = |E| \leq 3|V| - 6$, and the claimed upper bound can be derived by

$$\sum_{u \in V} \frac{|S(u)| - 1}{2} \leq \frac{3|V| - 6 - |V|}{2} = |V| - 3.$$

Moreover, the split procedure preserves the st-planarity of G. □

Notice that we did not prove a time bound. One may argue that the procedure described in the proof of Lemma 3.25 works in linear time, but for now we leave it this way and turn our attention to the next problem. Surely, the upper bound of $|V| - 3$ is a solid result, but we will push this a bit further from a practical point of view, and focus on the problem of finding a minimum set of edges to split. In the following, we will describe an algorithm that solves this problem in linear time, thereby, implying a linear-time bound for Lemma 3.25.

To do so, we introduce some more notation. As usual, let $u \in V$ be a vertex with successor list $S(u) = \{v_1, \ldots, v_m\}$. If we choose now a particular $1 \leq h \leq m$ at u, then we have to split every edge (u, v_{i+1}) with $i < h$ for which there exists a path $v_{i+1} \rightsquigarrow v_i$ and every edge (u, v_i) with $h \leq i$ for which G contains a path $v_i \rightsquigarrow v_{i+1}$. Let the number of these edges be $L(u, h)$ and $R(u, h)$, respectively. More specifically, we define $L(u, h)$ to be

$$L(u, h) = |\{i < h : v_{i+1} \rightsquigarrow v_i\}|$$

and, in a symmetric manner, $R(u, h)$ to be

$$R(u, h) = |\{h \leq i : v_i \rightsquigarrow v_{i+1}\}|.$$

Minimizing the total number of edges that have to be split, for which we have already an upper bound due to Lemma 3.25, can therefore be expressed by

$$\sum_{u \in V} \min_{1 \leq h \leq m} L(u, h) + R(u, h) \leq |V| - 3.$$

From an algorithmic point of view, we are interested in the value of h at every u such that the sum $L(u, h) + R(u, h)$ is minimized. For the sake of a shorter algorithm, let us put some more effort into this by rewriting $L(u, h)$ in recursive form:

$$L(u, h) = \begin{cases} 0 & \text{if } h = 1 \\ L(u, h-1) + 1 & \text{if } v_h \rightsquigarrow v_{h-1} \\ L(u, h-1) & \text{otherwise} \end{cases}$$

Similarly, we may rewrite $R(u,h)$ as

$$R(u,h) = \begin{cases} R(u) & \text{if } h = 1 \\ R(u,h-1) - 1 & \text{if } v_{h-1} \rightsquigarrow v_h \\ R(u,h-1) & \text{otherwise} \end{cases}$$

with $R(u,1) = R(u) = |\{1 \leq i < m : v_i \rightsquigarrow v_{i+1}\}|$ being the total number of consecutive successors for which there exists a path of the form $v_i \rightsquigarrow v_{i+1}$. And as result, we can now express the sum $L(u,h) + R(u,h)$ recursively by

$$L(u,h) + R(u,h) = \begin{cases} R(u) & \text{if } h = 1 \\ L(u,h-1) + R(u,h-1) + 1 & \text{if } v_h \rightsquigarrow v_{h-1} \\ L(u,h-1) + R(u,h-1) - 1 & \text{if } v_{h-1} \rightsquigarrow v_h \\ L(u,h-1) + R(u,h-1) & \text{otherwise.} \end{cases}$$

As a next step, we shift this sum by $R(u)$. More specifically, we define $C(u,h) = L(u,h) + R(u,h) - R(u)$. Notice that $R(u)$ does not depend on h, therefore, the following holds:

$$\sum_{u \in V} \min_{1 \leq h \leq m} L(u,h) + R(u,h) = \sum_{u \in V} \min_{1 \leq h \leq m} C(u,h) + \sum_{u \in V} R(u)$$

Clearly, it suffices to minimize $C(u,h)$ for every u. Moreover, we can express $C(u,h)$ in a similar way as the recursive variant of $L(u,h) + R(u,h)$:

$$C(u,h) = \begin{cases} 0 & \text{if } h = 1 \\ C(u,h-1) + 1 & \text{if } v_h \rightsquigarrow v_{h-1} \\ C(u,h-1) - 1 & \text{if } v_{h-1} \rightsquigarrow v_h \\ C(u,h-1) & \text{otherwise} \end{cases} \tag{3.6}$$

It is not difficult to see now that the value of h for which $C(u,h)$ is minimum, can be found by a single pass over the successor list $S(u)$. We already discussed how to test for the existence of paths. By borrowing parts of the algorithm presented earlier, a description of an algorithm that determines an optimum set of edges to split, becomes straightforward.

3.4.2 An optimal linear-time transformation

We assume that an embedded planar st-graph $G = (V,E)$ is given and the objective is to compute a set of edges E_{split} with minimum cardinality such that after splitting every edge in E_{split}, the resulting graph admits a bitonic st-ordering. For the algorithm, we follow roughly the same approach taken

```
procedure BitonicMinSplit
begin
    E_split ← ∅;
    for u ∈ V with S(u) = {v_1, ..., v_m} do
        h ← 1;
        c_min ← c ← 0;
        for i = 2 to m do
            w ← faceSink(u, v_{i−1}, v_i);
            if w = v_{i−1} then c ← c + 1;
            if w = v_i then c ← c − 1;
            if c < c_min then
                c_min ← c;
                h ← i;
            end
        end
        for i = 1 to h − 1 do
            if v_i = faceSink(u, v_i, v_{i+1}) then
                E_split ← E_split ∪ (u, v_i);
        end
        for i = h to m − 1 do
            if v_{i+1} = faceSink(u, v_i, v_{i+1}) then
                E_split ← E_split ∪ (u, v_{i+1});
        end
    end
    return E_split
end
```

Algorithm 10: Algorithm for computing the minimum set of edges to split of an embedded planar st-graph $G = (V, E)$ with its successor lists given.

in Algorithm 9, that is, traversing the successor list of every vertex u and inspecting the sink of the corresponding face to test for the existence of paths between two consecutive successors of u. A full listing of the procedure is given in Algorithm 10.

Based on the recursive expression given earlier in Equation 3.6, we maintain a counter that during the traversal, say at v_i, holds the value $C(u, i)$. More specifically, we test if either v_{i-1} or v_i is the sink of the common face of u, v_{i-1} and v_i. In the former case, there exists then a path $v_i \rightsquigarrow v_{i-1}$, and from Equation 3.6 we get $C(u, i) = C(u, i-1) + 1$. Thus, we increment the counter, whereas when v_i is the sink, we decrement it. Now it is not difficult to see that Algorithm 10 computes $1 \leq h \leq m$ for which $C(u, h)$ is minimum.

Having determined h, we repeat the traversal by considering two cases. If

we find a path $v_{i+1} \rightsquigarrow v_i$ with $i < h$, then the edge (u, v_i) has to be split in order to destroy the path. In the other case, that is, when we encounter a path $v_i \rightsquigarrow v_{i+1}$ with $h \leq i < m$, we add (u, v_{i+1}) to E_{split}. The number of edges added to E_{split} by one traversal is exactly $L(u, h) + R(u, h)$ by construction. Since we have chosen h such that $C(u, h)$ is minimum, we added the minimum number of edges to E_{split}.

As a result Algorithm 10 provides us with a set of edges E_{split} to split such that we can run Algorithm 9 on the resulting graph and be sure that it will compute a bitonic *st*-ordering. At this point it should be mentioned that these two algorithms can be merged into one. The only detail to consider, is that the edge splits in Algorithm 10 must happen before the edge insertions of Algorithm 9. However, for the sake of a clear presentation we leave it this way and draw some conclusions instead. Since we argued already its proof, we state the following lemma without one.

▶ **Lemma 3.26.** *Every embedded planar st-graph $G = (V, E)$ can be transformed into a planar st-graph that admits a bitonic st-ordering by splitting every edge at most once. Moreover, the minimum number of edges to split is at most $|V| - 3$ and they can be found in linear time.*

Now we may use this result to create upward planar drawings of planar *st*-graphs with few bends.

▶ **Theorem 3.27.** *Every embedded planar st-graph $G = (V, E)$ admits an upward planar poly-line drawing within quadratic area having at most one bend per edge and at most $|V| - 3$ bends in total. Moreover, such a drawing can be obtained in linear time.*

Proof. We use Lemma 3.26 to obtain a new planar *st*-graph $G' = (V', E')$ with $|V'| \leq 2|V| - 3$ and $|E'| \leq |E| + |V| - 3$ and a corresponding bitonic *st*-ordering π. Then with the help of Theorem 3.15 that uses Algorithm 8, an upward planar straight-line layout for G' is computed. Replacement of the dummy vertices by bends, yields an upward planar poly-line drawing for G of size at most $(4|V| - 8) \times (2|V| - 4)$. □

Recall that by Theorem 2.1 every upward planar graph is a spanning subgraph of a planar *st*-graph. Since it is a spanning subgraph and not just any subgraph, the number of vertices does not increase. Therefore, the bound of $|V| - 3$ translates to all upward planar graphs.

▶ **Corollary 3.28.** *Every upward planar graph $G = (V, E)$ admits an upward planar poly-line drawing within quadratic area having at most one bend per edge and at most $|V| - 3$ bends in total.*

Notice that we did not bound the runtime. Augmenting an upward planar graph into a planar *st*-graph is not a trivial task, but can be accomplished using upward planarity testing algorithms. However, recall that upward planarity testing in the variable embedding scenario is *NP*-complete [49]. When the embedding is fixed, the problem becomes polynomial time solvable, and an augmentation to a planar *st*-graph is usually part of the upward planarity testing procedure, see for example [12, 35].

3.4.3 Experimental results

The experimental results of the recognition algorithm have shown that the amount of planar *st*-graphs that admit a bitonic *st*-ordering is very small. Let us now have a look how many edges are necessary in practice to transform these instances into graphs that admit a bitonic *st*-ordering.

We use the same instances for the benchmark as for the recognition experiments, including those that admit a bitonic *st*-ordering. Based on their density and size, we computed the number of edges to split, that is $|E_{\text{split}}|$. The average number of edge splits for a fixed size and density serves as the result. These results are shown in Figure 3.21 in which they are compared to the number of transitive edges. Recall that in the dominance drawing based approach of Di Battista et al. [39] these have to be split in order to obtain the required reduced planar *st*-graph and serve later as bend points. Additionally, the upper bound of $|V| - 3$ from Theorem 3.27 has been added.

The results clearly show that the advantage of having only to split one of the two transitive edges involved in a forbidden configuration is not only significant in theory. Especially for the sparser larger instances ($\delta = 1.5$ and $\delta = 2$) only a fraction (3% and 11%) of the upper bound $|V| - 3$ has to be split, which is about 10% and 16% of the transitive edges. For the denser instances ($\delta = 2.5$ and maximal planar), the effect is not as strong (22% and 34% of the upper bound), but at most about 20% of the transitive edges have to be split.

Figure 3.22 shows an example for an upward planar poly-line drawing created by our implementation. In order to transform the embedded planar *st*-graph into one that admits a bitonic *st*-ordering, three edge splits are necessary, each resulting in a bend.

3.5 Visibility & contact representations

Before drawing final conclusions for this chapter, we turn our attention to another application for bitonic *st*-orderings: *contact representations* using rectilinear T-shaped polygons. Instead of developing another incremental draw-

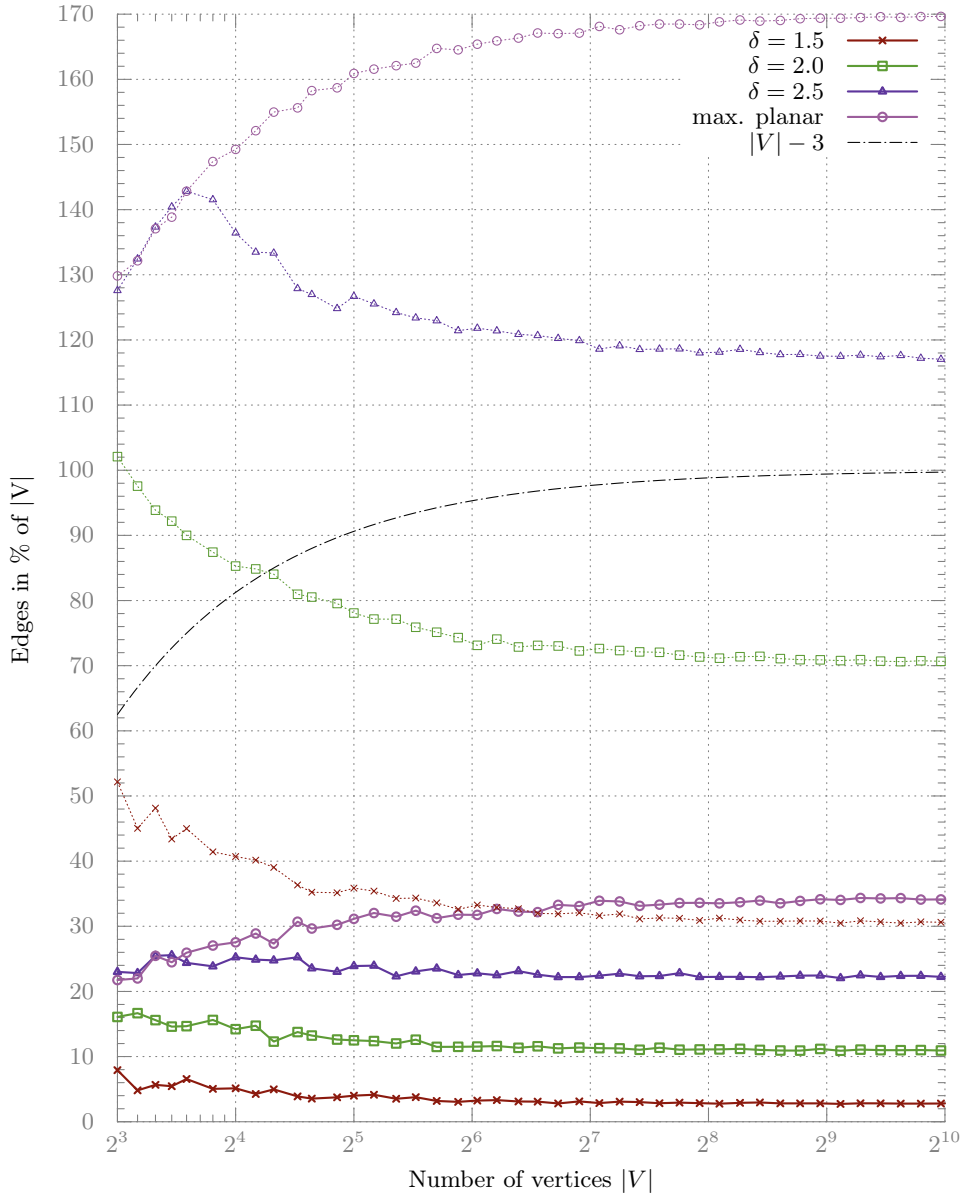


Figure 3.21. The average amount of edge splits (solid, thick) in comparison to the number of transitive edges (dotted, thin). The upper bound of $|V| - 3$ from Theorem 3.27 is drawn dashed-dotted.

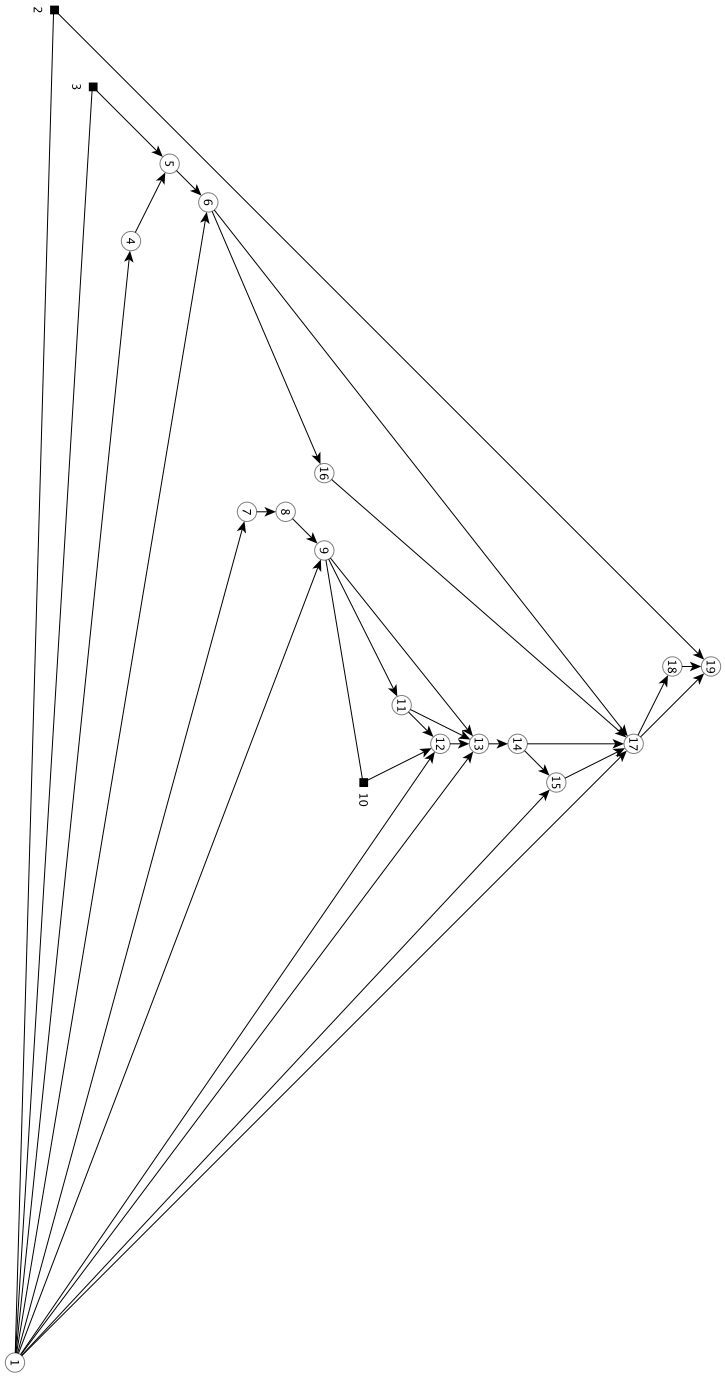


Figure 3.22. Example for an upward planar poly-line drawing of a planar *st*-graph $G = (V, E)$ with $|V| = 16$ and $|E| = 30$. Circles represent vertices of G, whereas squares indicate bends. The labels correspond to the rank in the bitonic *st*-ordering.

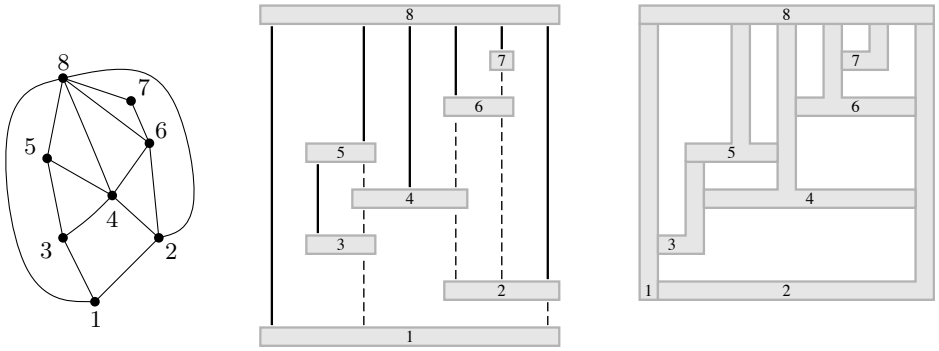


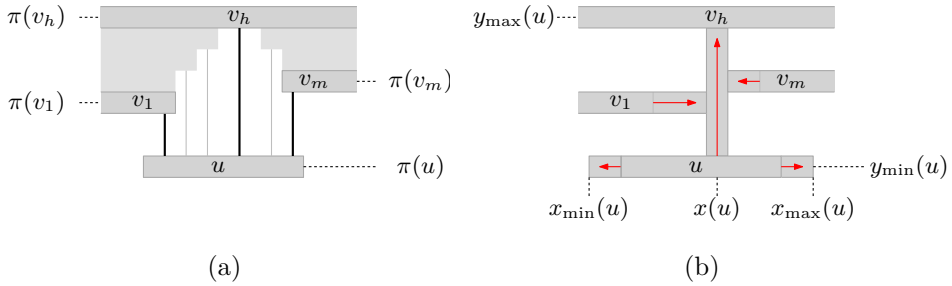
Figure 3.23. (a) Planar graph G and a bitonic st-ordering π. (b) Visibility representation for G when using π for the y-coordinates. The edges to the highest successor are drawn solid. (c) The resulting T-shaped contact representation.

ing algorithm, we use a visibility representation in conjunction with a bitonic st-ordering. This combination yields a special property that can easily be exploited, and we demonstrate this by describing how to obtain the aforementioned contact representation from it.

The idea of a rectilinear T-shaped contact representation is to represent a planar graph by touching sides of simple interior-disjoint polygons, in this case upside-down oriented T-shaped polygons. Alam et al. [2, 3] recently used these as an intermediate step to create cartograms. Their approach employs *Schnyder realizers* and their close relationship to canonical orderings. For more details see [2, 3]. Alam et al. have not been the first to construct such drawings, an earlier result is due to Sarrafzadeh and Yeap [73].

We assume that an embedded planar graph $G = (V, E)$ and a corresponding bitonic st-ordering π is given. The common way to obtain a visibility representation for G can be summarized as follows: The y-coordinates $y(v)$ of the horizontal segments that represent the vertices $v \in V$ of G are computed by an optimal topological numbering of the planar st-graph induced by an st-ordering. For the x-coordinate $x(e)$ of a vertical segment that represents an edge $e \in E$, the same procedure is repeated but on the dual planar st-graph. Since an optimal topological numbering can be obtained in linear time, a visibility representation can be obtained within the same time frame [35].

We skip the first step and choose π itself for the y-coordinates, that is, $y(v) = \pi(v)$. As a result every vertex has now its own row that corresponds



(a) (b)

Figure 3.24. (a) Successors $v_1, \ldots, v_h, \ldots, v_m$ of u whose ordering in the embedding is bitonic with respect to the y-coordinates. (b) Creating a pole at v_h, that is the highest successor, and pulling the bars of the remaining ones towards it.

to its rank in π. Figure 3.23b shows an example of the resulting visibility representation for the graph depicted in Figure 3.23a. Although a visibility representation can be derived this way for any st-ordering, we may now benefit from the property that π is a bitonic st-ordering. Since for every $u \in V$, $S(u)$ is bitonic with respect to π, by construction it is also bitonic with respect to the y-coordinates, that is, the successors are located above u in an increasing and then a decreasing staircase pattern. See Figure 3.24a for an illustration.

By using a simple trick, we now transform this wedge-like structure into a rectilinear T-shaped polygon. The idea is straightforward: We create a vertical segment on top of the horizontal bar that reaches all the way up to v_h, that is the highest successor of u. Afterwards we pull the bars of the remaining successors towards this pole. See the arrows in Figure 3.24b for a sketch of the idea. Notice that in case of a non-bitonic st-ordering, a single pole is not sufficient. The contact representation of the example is shown in Figure 3.23c.

Let us describe this idea in more detail. We denote by $x_{\min}(u)$ ($x_{\max}(u)$) the left (right) border of the upside-down T representing u, and $x(v)$ the horizontal offset of the pole. Furthermore, let $y_{\min}(u)$ and $y_{\max}(u)$ denote the vertical offset of the horizontal bar and the upper border of the pole, respectively. Then, for every $u \in V$ with $S(u) = \{v_1, \ldots, v_h, \ldots, v_m\}$ for which $y(v_1) < \cdots < y(v_h) > \cdots > y(v_m)$ holds, we create the vertical segment by choosing $x(u) = x(u, v_h)$, where $x(u, v_h)$ denotes the x-coordinate of (u, v_h) in the visibility representation. Furthermore, we set $y_{\max}(u) = y_{\min}(v_h)$. For the remaining successors v_i with $1 \leq i < h$, that is, those located to the left of the pole, we establish contact with the pole from the left by choosing $x_{\max}(v_i) = x(u)$. In a symmetric manner, we set $x_{\min}(v_i) = x(u)$ with $h < i \leq m$ for those successors that are located on the right. Notice that $x_{\min}(u)$ and $x_{\max}(u)$ are only defined

in the case in which there exists such a pole on both sides. Otherwise, we have to ensure that the horizontal bar of u covers at least the attaching poles from below. See v_3 and v_5 in Figure 3.23c for an example. Hence, for every u' with $u \in S(u')$ and $\pi(u) = \max_{v \in S(u')}\{\pi(v)\}$, that is, all u' for which u is the highest successor, we set $x_{\max}(u) = \max\{x(u), x(u')\}$ and $x_{\min}(u) = \min\{x(u), x(u')\}$.

It is not difficult to see that this approach takes linear time. Hence we may state the following:

▶ **Lemma 3.29.** *Given an embedded planar graph with a bitonic st-ordering, a rectilinear T-shaped contact representation can be obtained in linear time.*

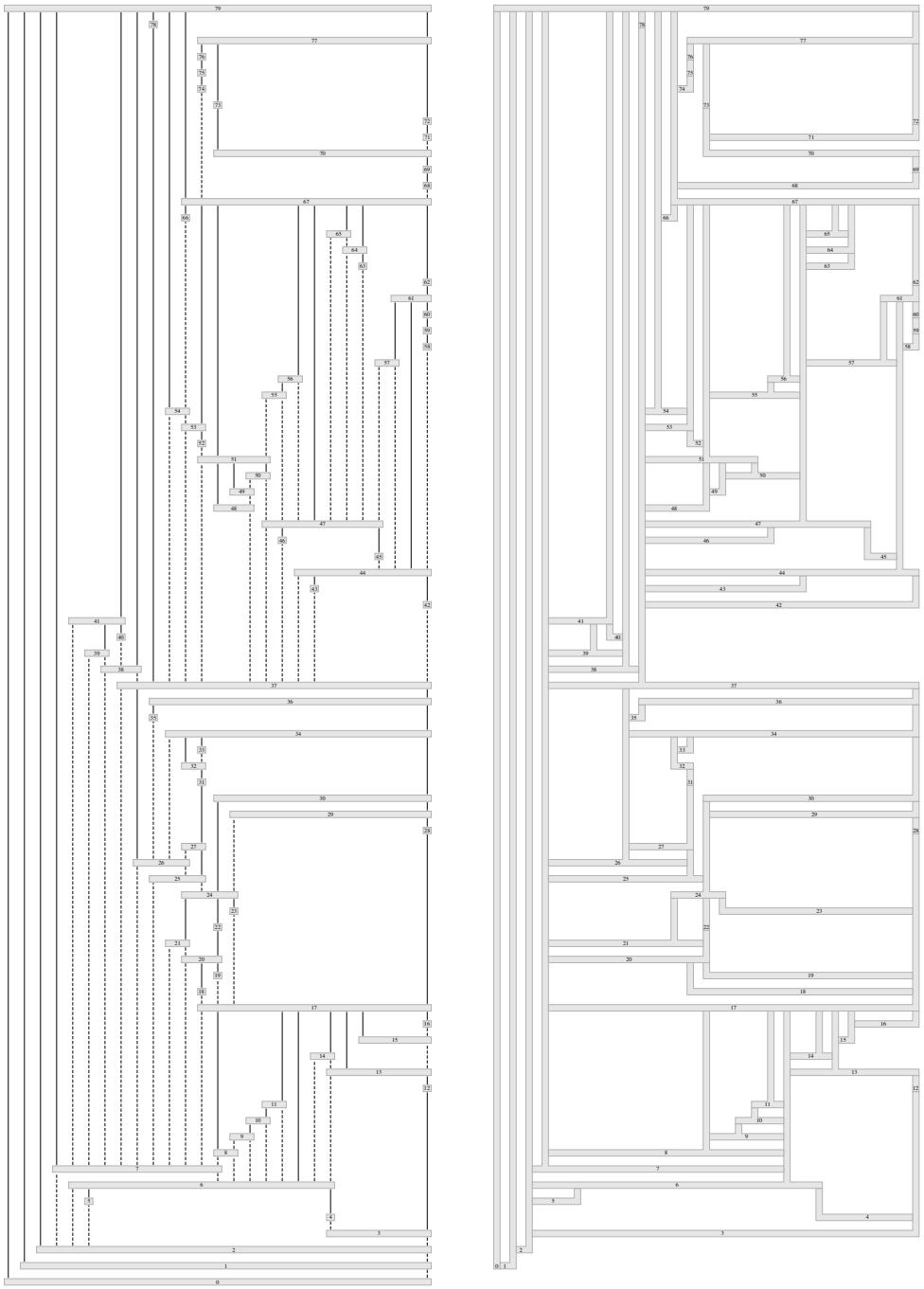
We implemented the above approach using the OGDF [65] with the algorithm for computing bitonic *st*-orderings for biconnected planar graphs. The output from our implementation for a larger graph is displayed in Figure 3.25. It shows the visibility representation and the obtained contact representation.

3.6 Conclusion

In this chapter we have introduced the concept of bitonic *st*-orderings, a special *st*-ordering with a property that is usually only available when using canonical orderings. We have shown that every undirected biconnected planar graph admits such an ordering. The proposed algorithm runs in linear time and utilizes SPQR-trees and canonical orderings. Unlike biconnected canonical orderings that preserve a given embedding, the presented algorithm may have to modify it. However, we presented a family of graphs for which such changes are necessary to establish the bitonic property, independently of the algorithm.

Afterwards, we described an adaptation of the planar straight-line drawing algorithm of de Fraysseix et al. By making only minor changes and without the need to modify key concepts, we demonstrated how easy the transition from a canonical ordering-based incremental drawing algorithm to one that uses the bitonic *st*-ordering can be.

The full strength of this approach is unleashed when applying the bitonic *st*-ordering to directed graphs. We gave a characterization of planar *st*-graphs that admit such a bitonic *st*-ordering, resulting in a linear-time algorithm that recognizes these and in case such an ordering exists, computes it. Although experiments on random planar *st*-graphs have shown that most of them do not admit a bitonic *st*-ordering, we may establish this property by splitting some edges. For this task a linear-time algorithm has been developed that computes the minimum set of edges to split. Based on this result, we were able to improve the upper bound on the number of bends required in any upward planar polyline drawing.



(a) (b)

Figure 3.25. (a) Larger example ($|V| = 80$, $|E| = 150$) of a visibility representation obtained from a bitonic *st*-ordering and (b) the corresponding contact representation.

One may argue now that the necessity of splitting edges in the directed case is a major drawback. However, one has to keep in mind that the bitonic *st*-ordering is essentially a canonical ordering that works for directed graphs and to our knowledge, the concept of canonical orderings has not been extended to directed graphs.

Moreover, every such attempt will face similar problems. Recall that we argued at the very beginning with the area requirements of upward planar straight-line drawings. Although we used this argument to derive a counterexample for a planar *st*-graph that does not admit a bitonic *st*-ordering, the core problem is that we are able to use the algorithm of de Fraysseix et al. for directed graphs without modifying its basic functionality. Of course, the bitonic *st*-ordering plays an important role for that, but just the fact that this algorithm works with our ordering on directed graphs, implies that we cannot handle all planar *st*-graphs due to the polynomial area argument for upward planar straight-line drawings. One may even take this one step further and claim that the algorithm of de Fraysseix et al. cannot work on all planar *st*-graphs without making major modifications to it, regardless of the type of ordering used, due to its polynomial area property.

This is a crucial point and has to be kept in mind when considering alternative orderings. At this point, we would like to point out that the advantage of both algorithms, the recognition and edge-split method, lies in their simplicity. Unlike the algorithm for undirected graphs that employs SPQR-trees, they consist only of a simple traversal of the embedded planar *st*-graph and do neither require sophisticated algorithms nor data structures. Basically, the pseudocode listings provided earlier can directly be used for an efficient implementation.

Open Problems

Let us now close this chapter with some open problems and ideas for future research. The first question that comes to mind is the case in which the graph is not biconnected such that it does not admit an *st*-ordering. This problem has not been addressed at all, but augmentation might be an option. The next problem is that the straight-line algorithm presented in Section 3.2.3 is a very basic one and does not utilize any improvements made to the canonical ordering based approach. See for example the work of Brandenburg [19] who improves the area to $\frac{4}{3}n \times \frac{2}{3}n$ or an earlier result due to Chobrak and Nakano [28]. But one has to keep in mind that such improvements do not necessarily carry over to upward planar straight-line drawings, due to different placement strategies.

When considering the larger example of the visibility representation and the corresponding contact representation in Figure 3.25, it becomes clear that using

the bitonic ordering for the y-coordinates leads to a drawing that is probably higher than necessary. It is not difficult to see that in this particular example there is much room for improving the area. This leads to the idea of introducing an optimal topological numbering with the bitonic property, because bitonicity provides the staircase pattern and that is all this approach requires.

With this in mind, the presented work already contains an idea that might be worth exploring. The obvious point to start from is the proof of Lemma 3.21 that serves as a basis for Algorithm 9. Recall that the basic idea is to augment the input graph G into a new planar st-graph G' such that at every vertex u there exists two paths, both directed towards a designated successor of u. These paths ensure that any st-ordering for G' is a bitonic one for G. It seems that this might work for any topological numbering. So maybe instead of computing an st-ordering for G' in Algorithm 9, one can compute an optimal topological numbering for G' which may result in a topological numbering for G whose visibility representation requires less area but still contains the staircase pattern.

However, recall that we gave an alternative characterization of bitonicity in Lemma 3.18 which assumes that the elements are pairwise distinct. This assumption is required for the proof. While by definition in an st-ordering the labels are distinct, this is usually not the case in a topological numbering. As a result one has to take the details into account when arguing along the proof of Lemma 3.21, since it uses implicitly this characterization via Proposition 3.20.

Another related property that might be used to tackle this problem is rather hidden. Recall that we have investigated the coordinate dependencies in the straight-line drawing algorithm. The result has been a very simple algorithm for computing the y-coordinates. It is not difficult to see that it can be easily adapted to work with a bitonic st-ordering. The obtained y-coordinates may serve themselves as a topological numbering of the vertices. The question arising is, whether this ordering has the bitonic property. Of course, this does not hold for any straight-line drawing, but maybe this is the case for drawings produced by this particular straight-line drawing algorithm. Even in the case in which the y-coordinates have the bitonic property, the corresponding ordering does not immediately result in a visibility representation of smaller height. But such a visibility representation has another property that is of interest: By construction and in a trivial way, there exists a transformation from the visibility representation into a straight-line drawing without changing the y-coordinates.

This leads us to another idea: In general the problem of transforming a visibility representation into a planar straight-line drawing while maintaining the y-coordinates is not a trivial task. Biedl [14, 15] discusses in her work various transformations. She shows in [15] how to transform a visibility representation into a planar straight-line drawing while preserving the y-coordinates. The ap-

proach, however, produces in the worst case a planar straight-line drawing with exponential width. The exponential width property is unavoidable since otherwise one would be able to compute a visibility representation for any planar st-graph and transform it into an upward planar straight-line drawing within polynomial area, and we already argued that this is impossible for some planar st-graphs. However, we may ask now if that works for the special visibility representation obtained from a bitonic st-ordering. More precisely, can every such visibility representation be transformed into a planar straight-line drawing within polynomial area that preserves the y-coordinates?

Last but not least, we would like to mention an open problem that lead to the development of the concept of bitonic st-orderings in the first place. In [5], we show that every 4-planar graph admits an octilinear drawing with one bend per edge. In an octilinear drawing, the line segments representing edges are restricted to be vertical, horizontal or diagonal. The approach taken in [5] uses a rather simple canonical ordering based algorithm for the triconnected case. However, the SPQR-tree approach that is used to generalize the idea to the biconnected case is quite involved and an implementation is not straightforward. This would be a perfect application for the bitonic st-ordering. However, an adaptation that is as simple as the straight-line algorithm seems unlikely.

The biggest problem to solve are the vertices with only one predecessor. These may have three successors, a case that is not covered by the triconnected algorithm described in [5]. However, in [5] we also present an algorithm for the 5-planar case that in difference to the 4-planar case may produce drawings that require exponential area. This algorithm might provide a solution to the problem of dealing with three successors, but some attention must be paid for the polynomial area property.

4 Two-page Book Embeddings of Bounded Degree Graphs

Book embeddings have a long history and arise in various application areas such as VLSI design, parallel computing, design of fault-tolerant systems and bioinformatics, see e.g. [17, 30, 43]. In a *book embedding* the placement of the vertices is restricted to a line, the *spine* of the book. The edges are assigned to different *pages* of the book. A page can be thought of as a half-plane bounded by the spine where the edges are drawn as circular arcs between their endpoints. We say that a graph admits a *k-page book embedding* or is *k-page embeddable* if one can assign the edges to k pages and there exists a linear ordering of the vertices on the spine such that no two edges of the same page cross. The minimum number of pages required to construct such an embedding is the *book thickness* or *page number* of a graph.

Embedding graphs in books with as few pages as possible has received much attention in the past. There is an extensive amount of literature on embedding various types of graphs into books; for an overview see e.g. [43]. The focus of this chapter lies on planar undirected graphs and, therefore, we concentrate on related work that is concerned with them. An important early work is due to Bernhart and Kainen [10] which coined the term book thickness and describes fundamental properties of book embeddings. They show bounds for the book thickness of various classes of graphs and offer a different perspective on book embeddings.

Usually a book embedding is considered to be some kind of linear layout, but one may also consider the vertices in a cyclic order. More specifically, if there exists a k-page book embedding for a graph $G = (V, E)$ with vertex ordering $v_1, \ldots, v_n$ and some fixed page assignment of the edges, then Bernhart and Kainen [10] observe that one may shift the vertices in a cyclic manner, such that $v_2, \ldots, v_n, v_1$ is still a feasible vertex ordering.

This property can also be explained in an intuitive way by an illustration. Instead of considering a drawing in which the vertices are drawn on a line representing the spine, we draw them on a circle according to the ordering stemming from the spine, whereas the edges are drawn inside the circle and colored according to the page assignment. It is not hard to see that this does not introduce any crossings among edges of the same color, because we basically

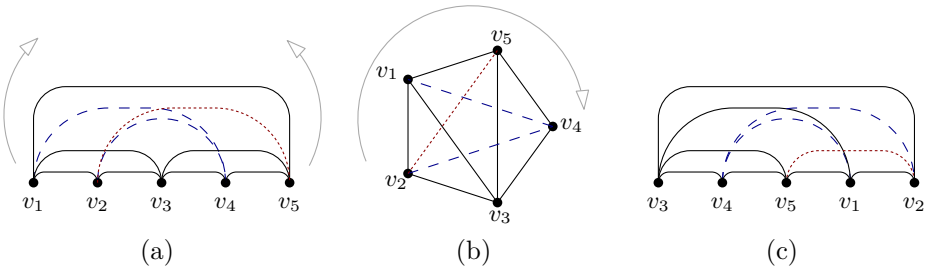


Figure 4.1. Example for cyclic shifting in a three-page book embedding. The page assignment is indicated by the line style (solid, dash, dotted). (a) Folding the 3-page book embedding of K_5 with vertex ordering $v_1, \ldots, v_5$ into a circular layout. (b) Cutting open the circular layout between v_2, v_3, and (c) unravelling it into a book embedding with vertex ordering v_3, v_4, v_5, v_1, v_2.

can bend the spine into a circle without changing the ordering of the vertices. Figure 4.1a shows a three-page embedding of K_5, which is turned into a circular drawing (Figure 4.1b). The observation to make is the following: Of course, we can reverse this process in order to obtain the initial book embedding, but instead of "cutting" the circle between v_1 and v_n again, we may choose a different position. Figure 4.1c shows the result when choosing v_2 and v_3 instead. Based on this circular layout, it immediately follows that the graphs that require only a single page, are exactly the outer planar graphs.

This property enables us to choose the first vertex in any book embedding, regardless of its book thickness. Recall that in the last chapter, we have been quite keen on the connectivity of the input graph. As we will see shortly, connectivity plays also a crucial role when it comes to graphs that admit a two-page book embedding. But before we turn our attention to this problem, we discuss the following result that does not depend on the book thickness, and uses the aforementioned property of cyclic shifts. It enables us to focus on biconnected graphs instead of the more general case.

▶ **Lemma 4.1.** *[10] The book thickness of a graph is the maximum book thickness of its biconnected components.*

It is not hard to see that we may assume that the graph is connected, because otherwise we may just line up the book embeddings of the connected components. A proof to get from the connected to biconnected case can be found in various places. Bernhart and Kainen [10] present one in their early work. A more algorithmic approach is given in the thesis of Heath [55]. The idea is constructive and yields a linear-time algorithm for reassembling the book embeddings of the biconnected components.

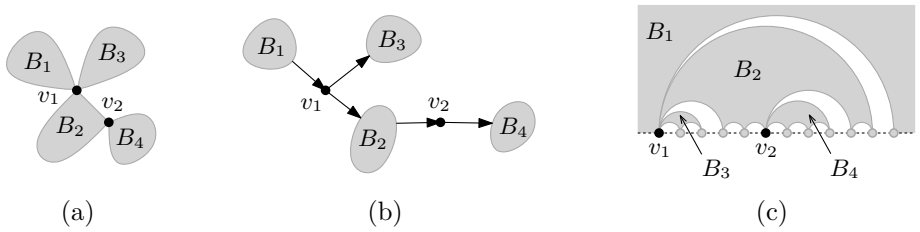


Figure 4.2. (a) Connected graph with two cut vertices v_1, v_2 and four biconnected components $B_1, \ldots, B_4$. (b) The corresponding BC-tree rooted at B_1. (c) The resulting book embedding after shifting and composing the ones of the biconnected components.

One may summarize such an algorithm as follows. We decompose a graph $G = (V, E)$ into its biconnected components $B_1, \ldots, B_m$ using a BC-tree (Figure 4.2a and 4.2b). Now assume that for every biconnected component B_i with $1 \leq i \leq m$, we have a book embedding given that requires $\mathrm{bt}(B_i)$ pages. We have already seen that besides the linear ordering of the vertices on the spine, one may also consider them in a cyclic order. This property can be exploited in the following sense: We root the BC-tree at an arbitrary B-node, say B_1, thus, inducing a hierarchy on the cut vertices. Figure 4.2b shows an example. Hence, every biconnected component B_i with $1 < i \leq m$ contains exactly one cut vertex that represents its parent, and every cut vertex has exactly one parent biconnected component.

In the corresponding book embedding of B_i, we may shift the vertices on the spine in a cyclic manner such that the parent cut vertex is the first. The result is still a book embedding occupying $\mathrm{bt}(B_i)$ pages, but we are now able to nest them according to the hierarchy as implied by the BC-tree. To do so, we follow a simple principle: The book embeddings of the children of a cut vertex v_c are being placed directly to the right of v_c in the book embedding of v_c's parent.

The idea is illustrated in Figure 4.2c in which for example B_4 has been placed directly to the right of v_2 and its successor on the spine in the book embedding of B_2. In a similar manner, B_2 and B_3 use the position of v_1 in B_1 as nesting position. Since two biconnected components have at most one vertex in common, we can always nest them such that they only overlap at the cut vertices, thus, not requiring any additional pages. It follows then that G requires at most as many pages as the book embedding of the biconnected component with the most pages, that is, for the book thickness of G holds $\mathrm{bt}(G) = \max_{1 \leq i \leq m}\{\mathrm{bt}(B_i)\}$. It is not hard to see that this technique does not extend to separation pairs, because they are not necessarily consecutive on the spine, but it has the advantage that it works for any number of pages.

The book thickness of planar graphs has received much attention in the past. Especially, in the eighties a lot of work has been done on this topic. Starting with Bernhart and Kainen [10] who were about the first to discuss the number of pages required for planar graphs. They conjectured that the book thickness of a planar graph can be arbitrarily large. We will see shortly that some planar graphs require at least three pages. However, in 1984, Buss and Shor showed that nine pages suffice and within the same year, Heath [54] presented an algorithm that tightens this result to seven pages. Five years later, Yannakakis [72] described a linear-time algorithm to embed every planar graph into a book of four pages which still stands today. It should be mentioned that in a preliminary version [71] of [72], it was claimed that four pages are also necessary. However, a proof has never been given, and the problem of whether the bound of four pages is tight or not, is still considered to be an open problem.

In the remainder of this chapter, we focus on two-page book embeddings, more specifically, we study the problem of embedding 3-planar graphs and, afterwards, 4-planar graphs into books with two pages. But before we turn our attention to the bounded degree problems, we first deal with the related work on two-page embeddable graphs. Afterwards, we focus on triconnected 3-planar graphs. Of course, these are covered by the 4-planar case and by a result from Heath [55] who already showed in the eighties that the 3-planar graphs have book thickness two. However, for the triconnected case, we describe a simple canonical ordering-based algorithm. Afterwards, we turn our attention to the 4-planar case. We show that every triconnected 4-planar graph is two-page embeddable and a corresponding ordering can be found in linear time. Before drawing final conclusions, a short summary of the general 4-planar case is given. However, the general case is not part of this thesis, and the summary serves only for reasons of completeness. Let us get started with the introduction of subhamiltonicity, and its relation to graphs with book thickness two.

4.1 Two-page book embeddings & subhamiltonicity

Book embeddings with two pages have found various applications in the field of graph drawing and visualization [41], which stems from the fact that a two-page book embedding is a special form of planar embedding having by definition a quite useful property: The vertices of a two page embeddable graph can be arranged on a line representing the spine, while each edge can entirely be drawn in one of the two half planes defined by this line without crossing another edge or the line. Clearly, such a drawing is planar, therefore, planarity is a necessary condition for a graph to have a book thickness of two.

Planarity, however, is far from being sufficient. In fact two-page book embeddings have a strong relation to Hamiltonian cycles in planar graphs. Recall that a Hamiltonian cycle is a cycle that visits every vertex exactly once. Bernhart and Kainen [10] show that a graph admits a two-page book embedding, if and only if it is a subgraph of a *planar Hamiltonian* graph, that is, a planar graph that contains a Hamiltonian cycle. We refer to this subset of the planar graphs as the *subhamiltonian* graphs. Notice that a subhamiltonian graph does not necessarily contain a Hamiltonian cycle. But since it is a subgraph of a planar Hamiltonian graph, one may add the missing edges without destroying planarity. In other words, a subhamiltonian graph admits a cyclic ordering of the vertices such that when adding the missing edges, the graph remains planar and the ordering is a Hamiltonian cycle. We refer to such a cyclic ordering as *subhamiltonian cycle*. One may think of a subhamiltonian cycle as a kind of Hamiltonian cycle that instead of solely using edges, may also cross faces.

The relation between two-page book embeddings and subhamiltonian cycles is based on the crucial observation that a subhamiltonian cycle in a planar graph, like a Hamiltonian cycle, partitions the edges into whether they are contained inside the cycle or not. While this immediately yields a feasible page assignment for the edges, the linear ordering of the vertices on the spine is obtained by breaking the cyclic ordering at an arbitrary position. Recall the circular drawing of K_5 in Figure 4.1b. Besides having only two pages, the only difference is that the circle representing the subhamiltonian cycle contains one page on the inside and one on the outside.

The general problem of determining if a graph is Hamiltonian is known to be NP-complete [48]. Unlike other problems that are NP-complete in general, but can be solved in polynomial time on planar graphs, the Hamiltonian cycle problem remains NP-complete in the planar case. Widgerson [70] shows that deciding whether a maximal planar graph contains a Hamiltonian cycle is NP-complete. Moreover, the same holds for the problem of determining if a planar graph is a subgraph of a planar Hamiltonian graph [70]. This immediately yields NP-completeness of two-page book embeddabilty.

This leads us to the question of which subsets of planar graphs are subhamiltonian. An early important result is due to Whitney [69], who proves that every maximal planar graph with no separating triangles is Hamiltonian (recall that a separating triangle is a 3-cycle whose removal disconnects the graph). Tutte [68] shows that every 4-connected planar graph has a Hamiltonian cycle. These two results are related in the following sense: Chen [21] for example gives a proof that every maximal planar graph with at least five vertices and no separating triangles is 4-connected. In order to obtain such a cycle, one may use the linear-time algorithm of Chiba and Nishizeki [23].

Clearly, in the maximal planar case, separating triangles play an important role due to their absence being a sufficient condition for Hamiltonicity. However, their existence does not necessarily imply that a maximal planar graph is not Hamiltonian, thus, requiring three pages in a book embedding. For the case in which there exists exactly one separating triangle in a maximal planar graph, Chen [21] shows that one may still obtain a Hamiltonian cycle. The idea is to decompose the problem at the triangle, compute two separate Hamiltonian cycles for the subproblems and merge them afterwards. We will use this idea later in the 4-planar case. Helden [56] improves this result further to two separating triangles. For more results on hamiltonicity of maximal planar graphs and triangulations, we refer the reader to the thesis of Helden [57], which also contains a proof that maximal planar graphs with five separating triangles are still Hamiltonian.

For subhamiltonicity the importance of separating triangles is not restricted to the maximal planar case. The following result due to Kainen and Overbay completely lifts this restriction in terms of connectivity.

▶ **Theorem 4.2.** *[60] Every planar graph without separating triangles is subhamiltonian.*

The proof of this result in [60] contains a quite elegant two-stage procedure: At first they show that a triconnected planar graph without separating triangles can be augmented into a maximal planar graph with the same property. Since we require this step later in the 4-planar case (Section 4.3), we prove here the following slightly stronger result which is concerned with the ability to cross a face. A subhamiltonian cycle H *crosses* a face, if there are two consecutive vertices in H that are incident to the face but not adjacent to each other.

▶ **Lemma 4.3.** *Every triconnected planar graph with no separating triangles has a subhamiltonian cycle that crosses every face at most once and it can be computed in linear time.*

Proof. In the triconnected case, Kainen and Overbay [60] construct a new maximal planar graph $G' = (V', E')$ by inserting a vertex into each non-triangular face of G and connect it to the vertices of that face (Figure 4.3a). Clearly this takes linear time. One may then argue with triconnectivity that this procedure does not introduce additional separating triangles. Figure 4.3b illustrates this argument. G' is maximal planar, free of separating triangles, hence, 4-connected. We can use the linear-time algorithm of Chiba and Nishizeki [23] to obtain a Hamiltonian cycle H' for G'. Deleting the newly inserted vertices $V' - V$ yields a subhamiltonian cycle H for G that crosses each face at most once. □

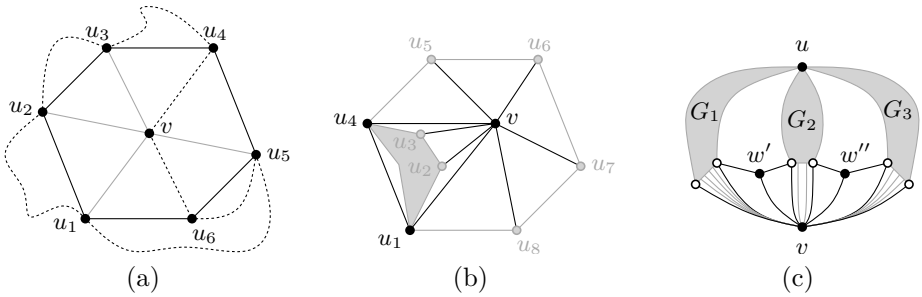


Figure 4.3. (a) A Hamiltonian cycle (dotted) in the augmented graph G' passing through a face of G via the inserted dummy vertex v in G' (b) A separating triangle introduced by stellating the face $u_1, \ldots, u_8$ implying a separation pair $\{u_1, u_4\}$ in the original graph G. (c) Eliminating a separation pair $\{u, v\}$ by inserting degree three vertices w', w'' between components G_1, G_2, G_3 at v to connect them.

The second step in the proof is an augmentation from biconnected to triconnected planar graphs without introducing separating triangles. Kainen and Overbay describe a technique that assumes a fixed embedding and then considers one vertex v of a separation pair $\{u, v\}$. The idea is to eliminate the pair by connecting specific neighbors of v to establish triconnectivity between components. Figure 4.3c illustrates the idea. The additional vertices between components are necessary to prevent the introduction of a separating triangle. Due to Lemma 4.1 one may assume biconnectivity, therefore, the claim follows.

This result is quite powerful as we will see in the next section that is concerned with 3-planar graphs, but the second step has a drawback, which will become evident in the 4-planar case. But let us get started with 3-planar graphs.

4.2 Two-page book embeddings of 3-planar graphs

We start with considering the case in which a 3-planar graph is given. Since K_4 is 3-planar but not outerplanar, it follows that two pages are necessary. In fact two pages are also sufficient as Heath shows. In his thesis [55], he describes a linear-time algorithm for embedding any 3-planar graph into a book of two pages. We begin by stating this result right away.

▶ **Theorem 4.4.** *[55] Every 3-planar graph is subhamiltonian.*

The key component of the algorithm is a special face traversal for biconnected 3-planar graphs (Heath refers to 3-planar as trivalent graphs). The idea is to construct a subhamiltonian cycle in an incremental manner based on a step-by-step traversal of the faces. The proposed method is quite involved, but uses a

concept that should sound familiar to the reader. Incrementally traversing the faces is essentially what a canonical ordering does. A quick look at the year in which Heath wrote his thesis (1985) reveals that it predates the introduction of canonical orderings. The idea now is to use a canonical ordering instead of this special traversal. This is only interesting from a technical point of view, because an implementation for a canonical ordering is most likely to be available compared to the traversal of Heath.

We proceed now in two steps: First, we describe a simple canonical ordering-based algorithm for the triconnected case. It uses the canonical ordering of Kant as described in Definition 3.2 to construct a subhamiltonian cycle for a triconnected 3-planar graph in an incremental manner. Afterwards, we give an alternative separating triangle-based proof for the result of Heath that is based on Theorem 4.2. One may argue that this section does not provide any new results, however, the algorithm is simple and the alternative proof afterwards introduces some ideas that are similar to the ones in the next section.

Canonical orderings of triconnected 3-planar graphs have some very special properties that have been used in the field of graph drawing extensively [9, 61]. Assume a triconnected 3-planar graph $G = (V, E)$ is given. Since G is 3-planar and triconnected, for every $v \in V$, $\deg(v) = 3$ must hold. Now let $V_1 \cup \cdots \cup V_K$ be a canonical ordering as defined by Kant, see Definition 3.2. From this definition, it follows that every vertex $v \in V_k$ with $1 < k < K$ has at least two neighbors in G_k and at least one in $G - G_k$. Since $\deg(v) = 3$ holds, we may assume that v has exactly two neighbors in G_k and exactly one in $G - G_k$. Notice that this holds, regardless of V_k being a singleton or a chain. Furthermore, $V_1 = \{v_1, v_2\}$ and $V_2 = \{v_3, \ldots\}$ holds. From the fact that G_2 is a simple cycle in which v_2 and v_3 are the neighbors of v_1, it follows that these two, together with v_n, are the only three neighbors of v_1. Of course, this assumes that $v_n \neq v_3$ holds, because otherwise $K = 2$ and G is triangle, thus, Hamiltonian in a trivial way.

The overall idea of our algorithm is as follows. Similar to a canonical ordering-based drawing algorithm, we add step-by-step the partitions, but instead of maintaining an invariant for a drawing, we maintain one for a subhamiltonian cycle. The intuition of this invariant is that when we consider a partition, then we may assume that the cycle enables us to pick it up from the contour and reroute it through the vertices of the partition. In order to achieve this, we have to ensure that the cycle surfaces on the outer face of G_k at specific vertices.

Let us describe this in a more formal manner. We follow the notation of the straight-line algorithms in Section 3.2, that is, we use $C_{k-1} = \{w_1, \ldots, w_m\}$ to denote the contour of G_{k-1}. Recall that C_{k-1} is the outer face of G_{k-1} without the edge (v_1, v_2). Hence, $w_1 = v_1$ and $w_m = v_2$ holds. Furthermore, let w_l and w_r be the leftmost and rightmost neighbor of V_k on C_{k-1}, respectively.

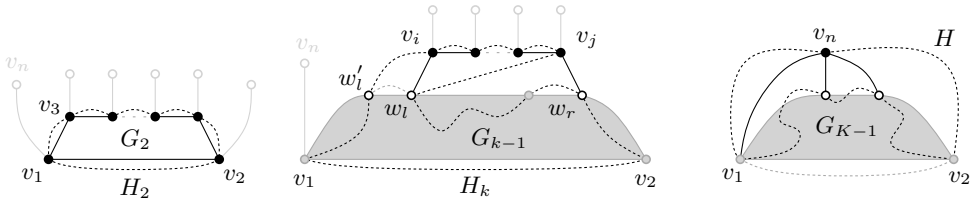
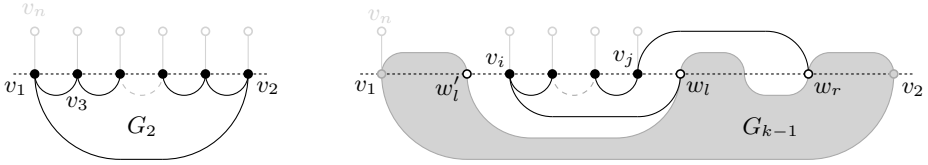


Figure 4.4. Augmenting a subhamiltonian cycle (dotted) in a triconnected 3-planar graph using a canonical ordering $V_1 \cup \cdots \cup V_K$. Edges that are not present yet are drawn gray. (a) Initial cycle H_2 for G_2 consisting of $V_1 = \{v_1, v_2\}$ and $V_2 = \{v_3, \ldots\}$ inducing a simple cycle. (b) Augmenting H_{k-1} into H_k when placing $V_k = \{v_i, \ldots, v_j\}$ by inserting v_i in between w'_l and w_l replacing the dotted gray part between w'_l and w_l. (c) The final cycle H after inserting $V_K = \{v_n\}$ in which v_1 and v_n appear consecutively.

Invariant: For every $3 \leq k \leq K$, we maintain a subhamiltonian cycle H_k for G_k. With a slight abuse of notation, we refer with $G_k \cup H_k$ to the Hamiltonian planar graph that results from adding the missing edges from H_k to G_k. Besides being subhamiltonian, H_k must satisfy the following condition: For every $v \in C_k$ that has a neighbor in $G - G_k$, v is on the outer face of $G_k \cup H_k$. Furthermore, the vertex directly preceding v in H_k is also on the outer face of $G_k \cup H_k$.

Base case: We start with $k = 2$ in which G_2 is a simple cycle consisting of $v_1, v_3, \ldots, v_2$. This cycle serves also as initial subhamiltonian cycle H_2, that is, we set $H_2 = \{v_1, v_3, \ldots, v_2\}$. See Figure 4.4a for an illustration of the initial cycle. Since $G_2 \cup H_2 = G_2$, the invariant holds.

Intermediate step: We show how to handle the k-th partition $V_k = \{v_i, \ldots, v_j\}$ with $2 < k < K$. Notice that we cannot assume that $v_i \neq v_j$ holds. Let w_l and w_r be the leftmost and rightmost neighbor of V_k on C_{k-1}, respectively. By our invariant, we may assume that w_l and the predecessor of w_l on H_{k-1}, say w'_l, are both on the outer face of $G_{k-1} \cup H_{k-1}$ and on C_{k-1}. We augment H_{k-1} to obtain H_k by inserting the vertices $v_i, \ldots, v_j$ of V_k in between w'_l and w_l. Figure 4.4b illustrates the procedure. Since $G_k \cup \{(w'_l, v_i), (v_j, w_l)\}$ is planar, H_k is a subhamiltonian cycle. It remains to show that H_k full fills the additional requirement. It is not hard to see that $v_i, \ldots, v_j$ are on the outer face of $G_k \cup H_k$ and so are their predecessors $w'_l, v_i, \ldots, v_{j-1}$ on H_k. However, for $w_l, \ldots, w_r$ this is not the case anymore. From the definition of canonical ordering, it follows that $w_{l+1}, \ldots, w_{r-1}$ do not have a neighbor in $G - G_k$ anyway. For w_l and w_r we may argue now that they both have already degree three in G_k, thus, there cannot exist another neighbor in $G - G_k$.



(a) (b)

Figure 4.5. (a) Corresponding book embedding of G_2 from Figure 4.4a. The spine that represents the subhamiltonian cycle is drawn dotted. (b) Placing $V_k = \{v_i, \dots, v_j\}$ in between w'_l and w_l, which corresponds to augmenting H_{k-1} into H_k as in Figure 4.4b.

This argument is by the way the reason why this approach does not extend to graphs of higher maximum degree. But in the 3-planar case, H_k complies with the invariant.

Last step: It remains the case $k = K$ in which we have to deal with $V_K = \{v_n\}$. Here we proceed exactly as in the case with $2 < k < K$ even though v_n has three neighbors. Recall that the leftmost neighbor of v_n is always v_1 and the predecessor of v_1 on H_{K-1} is v_2 due to the initialization. Hence, augmenting H_{K-1} into H_K by inserting v_n in between v_1 and v_2 yields a subhamiltonian cycle $H = H_K$ for G in which v_1, v_2, v_n appear consecutively (Figure 4.4c).

Although the described algorithm is solely based on augmenting a subhamiltonian cycle, one may also think of this procedure as incrementally building a two-page book embedding. Let us have a quick look at what happens during the steps in the corresponding book embedding. For the linear ordering of the vertices we choose v_1 to be the first and v_2 to be the last. Notice that they appear consecutively in H_k for every $2 \leq k < K$. For the page assignment of the edges, we consider the cycle clockwise and agree that every edge outside of it is to be drawn in the upper half plane. See Figure 4.5a for the initial book embedding of the simple cycle G_2. The placement of a partition V_k with $2 < k < K$ is illustrated in Figure 4.5b. The invariant guarantees that to the left of a vertex that has neighbor in $G - G_k$, the upper half plane is not used. Therefore, there is enough room on the spine to embed the corresponding vertices and edges.

In the previous section, we have seen that separating triangles are an important structure when it comes to subhamiltonicity. It is not hard to imagine that the possibilities for separating triangles in 3-planar graphs are rather limited, because each triangle is a cycle already requiring two edges from the possible three edges incident to a vertex. The next lemma confirms this claim and shows that biconnectivity is sufficient for their absence.

▶ **Lemma 4.5.** *A biconnected 3-planar graph is free of separating triangles.*

Proof. Let $G = (V, E)$ be a biconnected 3-planar graph and assume that there exists a separating triangle T in G such that the removal of T disconnects G into two subgraphs G' and G''. For the argument the embedding is not important. Since G is biconnected, there are at least two edges connecting G' to two distinct vertices of T. By a symmetric argument, this holds also for G''. Hence, at least one of the three vertices of T, say v, has a neighbor in both G' and G''. Furthermore, v has another two neighbors in T, because T is a 3-cycle, and it follows that $\deg(v) \geq 4$, which contradicts that G is 3-planar. □

Now it is not difficult to see that with the help of Theorem 4.2, we are able to claim that every 3-planar is subhamiltonian. Furthermore, the proof of Lemma 4.5 suggests that 4-planar graphs may contain separating triangles. And this is indeed the case, but they have a very special structure which we will investigate and exploit in the next section.

4.3 Subhamiltonicity of triconnected 4-planar graphs

In this section we restrict ourselves to triconnected 4-planar graphs. To state the main result of this section, we proceed in a step-by-step manner. First we investigate the special properties of separating triangles in 4-planar graphs, then we use those to derive a solution for a single separating triangle. Unlike Chen [21] and Helden [56], we are able to extend our approach to an unbounded number of triangles by exploiting the degree restriction.

Before investigating the properties of separating triangles, we introduce some notation. Given an embedded triconnected 4-planar graph G with a fixed outerface and a separating triangle T with vertices $V(T) = \{A, B, C\}$, we denote the subgraph of G contained in T by $G_{in}(T)$ and the subgraph of G outside T by $G_{out}(T)$. We also denote $\overline{G}_{in}(T) = G - G_{out}(T)$ and $\overline{G}_{out}(T) = G - G_{in}(T)$. Since G is triconnected and 4-planar, every vertex of T has degree four and is adjacent to exactly one vertex in $G_{in}(T)$ and $G_{out}(T)$, respectively. We denote these with A_{in}, B_{in}, C_{in} and $A_{out}, B_{out}, C_{out}$, respectively (see Figure 4.6).

▶ **Lemma 4.6.** *Given a 4-planar triconnected graph G and a separating triangle $T = \{A, B, C\}$, then $A_{in}, B_{in}, C_{in} (A_{out}, B_{out}, C_{out})$ are pairwise distinct or all represent the same vertex.*

Proof. In the other case, where w.l.o.g. $A_{in} = B_{in} = v$ and $C_{in} \neq v$, there exists a separation pair (v, C) contradicting the triconnectivity of G. A symmetric argument applies to $A_{out}, B_{out}, C_{out}$. □

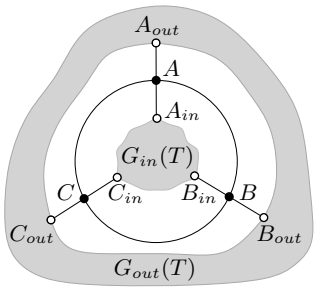


Figure 4.6. Triangle T separating $G_{in}(T)$ and $G_{out}(T)$ on removal.

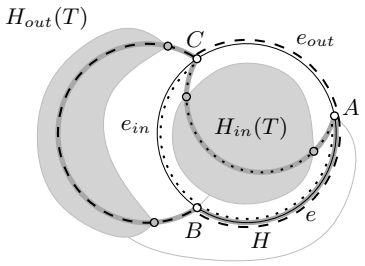


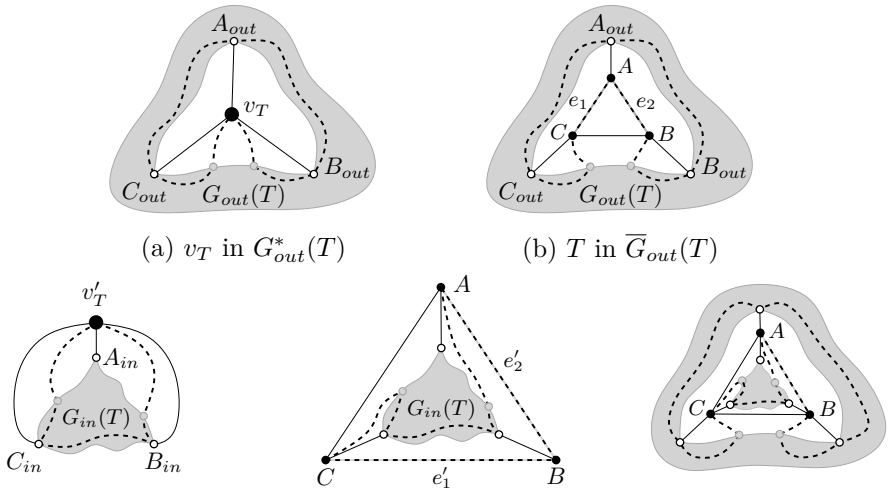
Figure 4.7. Merging $H_{in}(T)$ (dotted) and $H_{out}(T)$ (dashed) into H (bold gray).

▶ **Lemma 4.7.** *In a 4-planar triconnected graph, every pair of distinct separating triangles T and T' is vertex disjoint, that is, $V(T) \cap V(T') = \emptyset$ holds.*

Proof. Assume to the contrary that T and T' share an edge or a vertex. In the first case, let w.l.o.g. $e = (u, v)$ be the common edge and assume that T and T' are not nested. The degree of both u and v is at least five, since three edges are required for T, T' and two additional edges to connect $G_{in}(T)$ and $G_{in}(T')$ to T and T', respectively. Now suppose the two separating triangles are nested, and w.l.o.g. let T' be contained in T. Then in a similar manner, at u and v three edges are required for T, T' and two to connect $G_{out}(T)$ and $G_{in}(T')$ to T and T', respectively. In the second case, let v denote the common vertex. Regardless of T and T' being nested or not, v is part of two edge disjoint cycles which contribute each two edges to the degree of v. If w.l.o.g. T' is contained in T, then two additional edges are required for $G_{out}(T)$ and $G_{in}(T')$. Similar in the case in which they are not nested, $G_{in}(T)$ and $G_{in}(T')$ contribute an edge each. It follows in both subcases that $deg(v) \geq 6$. □

Consider now a 4-planar triconnected graph with a single separating triangle T. Similar to Chen [21], the idea is to compute two cycles $H_{in}(T)$ and $H_{out}(T)$ for $\overline{G}_{in}(T)$ and $\overline{G}_{out}(T)$ and link them via the separating triangle together. The crucial observation is that if two cycles intersect as illustrated in Figure 4.7, that is, they contain two edges of the triangle but have only one of them in common, then we can always merge them into one cycle.

▶ **Lemma 4.8.** *Let G be a triconnected 4-planar graph, T be a separating triangle, and $H_{in}(T)$ and $H_{out}(T)$ be two subhamiltonian cycles for $\overline{G}_{in}(T)$ and $\overline{G}_{out}(T)$, respectively. If $E(H_{in}(T)) \cap E(T) = \{e_{in}, e\}$ and $E(H_{out}(T)) \cap E(T) = \{e_{out}, e\}$ where $\{e, e_{in}, e_{out}\}$ are the edges of T, then G is subhamiltonian.*



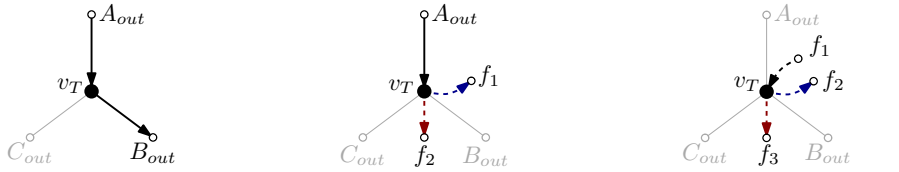
(a) v_T in $G^*_{out}(T)$ (b) T in $\overline{G}_{out}(T)$

(c) v'_T in $G^*_{in}(T)$ (d) T in $\overline{G}_{in}(T)$ (e) G with T and H

Figure 4.8. (a) Subhamiltonian cycle $H^*_{out}(T)$ in $G^*_{out}(T)$ containing v_T. (b) Augmenting $H^*_{out}(T)$ yields $H_{out}(T)$ containing edges $e_1 = (C, A)$ and $e_2 = (A, B)$. (c) Dummy vertex v'_T as replacement for T in $G^*_{in}(T)$ and a cycle $H^*_{in}(T)$. (d) Rerouting $H^*_{in}(T)$ through T resulting in $H_{in}(T)$ with edges $e'_1 = (C, B)$ and $e_2 = (A, B)$. (e) The result of merging $H_{in}(T)$ and $H_{out}(T)$ into a cycle H for G.

Proof. Let w.l.o.g. $e = (A, B)$, $e_{in} = (B, C)$ and $e_{out} = (A, C)$ as illustrated in Figure 4.7. The result of removing the edges of T from both cycles are two paths $P_{out} = B \rightsquigarrow C$ and $P_{in} = C \rightsquigarrow A$. Joining them at C and inserting e yields a subhamiltonian cycle. □

It remains to show that we can always find two cycles that satisfy the requirements of Lemma 4.8. In the following, we neglect the degenerated case of Lemma 4.6, where $G_{out}(T)$ or $G_{in}(T)$ is a single vertex, because finding a cycle in that case is trivial. Consider for example $\overline{G}_{out}(T)$, for $\overline{G}_{in}(T)$ a symmetric argument holds. To obtain $H_{out}(T)$, we temporarily replace T in $\overline{G}_{out}(T)$ with a single vertex v_T as depicted in Figure 4.8a. The resulting graph $G^*_{out}(T)$ remains 4-planar and triconnected, because $deg(v_T) = 3$ by construction and any path via T can use v_T instead. One may argue that this operation may introduce additional separating triangles. However, such a triangle must contain v_T and, therefore, $deg(v_T) = 4$, a contradiction. Now let us assume that $H^*_{out}(T)$ is a subhamiltonian cycle for $G^*_{out}(T)$. The idea is to reinsert T and reroute $H^*_{out}(T)$ through T such that the resulting cycle $H_{out}(T)$ contains two edges $e_1, e_2 \in E(T)$ as depicted in Figure 4.8b. In a symmetric manner, this approach can be applied for $H_{out}(T)$ as displayed in Figure 4.8c and 4.8d.



(a) Edge $\rightsquigarrow v_T \rightsquigarrow$ Edge (b) Edge $\rightsquigarrow v_T \rightsquigarrow$ Face (c) Face $\rightsquigarrow v_T \rightsquigarrow$ Face

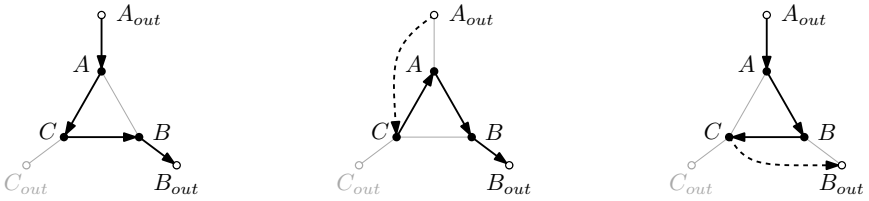
Figure 4.9. The three main cases at v_T: (a) The cycle uses two of the three edges incident to v_T. (b) The cycle enters via an edge and leaves through a face. (c) Predecessor and successor are not adjacent to v_T.

▶ **Lemma 4.9.** *Let G be a triconnected 4-planar graph and T be a separating triangle. Furthermore, let $G^*_{out}(T)$ denote the graph resulting from replacing T by a vertex v_T in $\overline{G}_{out}(T)$. A subhamiltonian cycle $H^*_{out}(T)$ for $G^*_{out}(T)$ can be augmented to a subhamiltonian cycle $H_{out}(T)$ for $\overline{G}_{out}(T)$ such that it contains two edges of T, that is, $E(H_{out}(T)) \cap E(T) = \{e_1, e_2\}$. If $H^*_{out}(T)$ crosses every face of $G^*_{out}(T)$ at most once, one may choose any pair $e_1, e_2 \in E(T)$ to lie on $H_{out}(T)$.*

Proof. In order to prove the claim, it is sufficient to consider every combination of e_1, e_2 and the location of the predecessor and successor of v_T in $H^*_{out}(T)$. In the following, we enumerate and describe in detail all possible cases that may occur when augmenting $H^*_{out}(T)$ such that the resulting cycle $H_{out}(T)$ contains two edges e_1, e_2 of T. However, to avoid any redundancies, we omit symmetric cases and consider for the same reason a directed cycle. We distinguish between three main cases depending on the relative location of the predecessor and successor of v_T in $H^*_{out}(T)$.

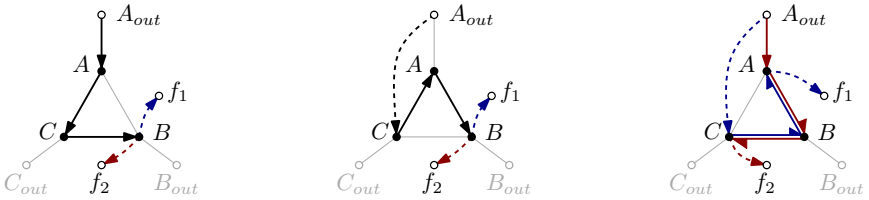
An overview of these main cases is shown Figure 4.9 and can be summarized as follows. The first case is depicted in Figure 4.9a in which the cycle contains two edges incident to v_T, whereas in the second case the cycle arrives via an edge but leaves through a face (Figure 4.9b). Here we have to distinguish between the location of the face, that is if the edge bounds the face or not (f_1 and f_2 in Figure 4.9b, respectively). In the third case, the cycle does not use any to v_T incident edges and visits it solely using faces (Figure 4.9c). Here again we have to distinguish between two sub cases based on if the cycle leaves v_T via the same face (f_2 in Figure 4.9c) as it arrives or not (f_3 in Figure 4.9c).

As a next step, we show that we can expand the separating triangle and reroute the cycle such that any pair of edges is part of the extended cycle. One exception is the case in which the cycle visits the vertex v_T via the same face ($f_1 \rightsquigarrow v_T \rightsquigarrow f_2$ in Figure 4.9c).



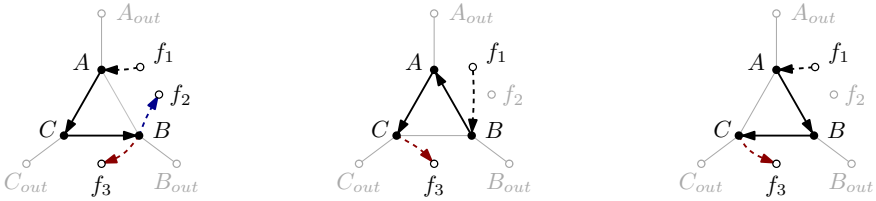
(a) (A, C) and (C, B) (b) (C, A) and (A, B) (c) (A, B) and (B, C)

Figure 4.10. (a) Dummy vertex v_T is replaced by the sequence A, C, B to obtain a cycle with the edges (A, C) and (C, B). (b) Sequence C, A, B yields a cycle with (C, A) and (A, B) where (A_{out}, C) requires it to cross a face. (c) Augmenting with A, B, C results in a cycle containing (A, B) and (B, C).



(a) (A, C) and (C, B) (b) (C, A) and (A, B) (c) (A, B) and (B, C) or (C, B) and (B, A)

Figure 4.11. In (a) and (b) the same sequences as before are used to obtain a cycle containing $(A, C), (C, B)$ and $(C, A), (A, B)$, respectively. Subcase-specific links are drawn in blue and red. (c) A more complicated case requiring one additional crossing of a face from A_{out} to C.



(a) (A, C) and (C, B) (b) (B, A) and (A, C) (c) (A, B) and (B, C)

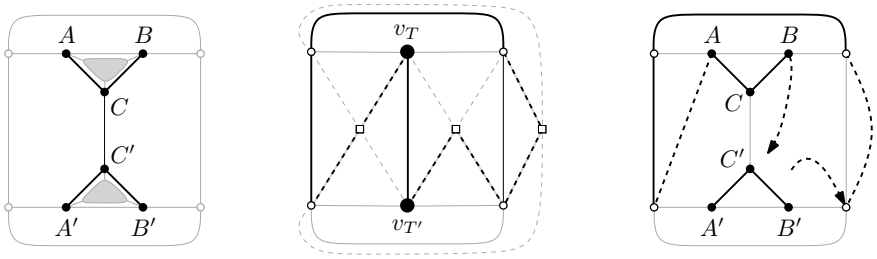
Figure 4.12. (a) Both subcases have a solution. (b,c) When the cycle uses two distinct faces ($f_1 \rightsquigarrow f_3$) a solution for both pairs of edges can be found. If only one face is used ($f_1 \rightsquigarrow f_2$), then no solution exists for the edges $(B, A), (A, C)$ and $(A, B), (B, C)$.

Case 1 (Edge $\rightsquigarrow v_T \rightsquigarrow$ Edge): Both the predecessor and successor of v_T in $H^*_{out}(T)$ are adjacent to v_T, hence, the cycle $H^*_{out}(T)$ contains two edges incident to v_T, let us say $(A_{out}, v_T), (v_T, B_{out})$ as illustrated in Figure 4.9a. Figure 4.10 depicts how $H^*_{out}(T)$ can be augmented such that every pair of edges of T is contained in $H_{out}(T)$. Notice that while for the pair $(A, C), (C, B)$ in Figure 4.10a no face crossing is required, for the two other pairs one additional face crossing is introduced (Figure 4.10b and 4.10c).

Case 2 (Edge $\rightsquigarrow v_T \rightsquigarrow$ Face): In this case, the predecessor, say A_{out}, is adjacent to v_T, while the successor is not. Since $H^*_{out}(T)$ is a subhamiltonian cycle, the successor is incident to one of the three faces incident to v_T. To cover all possible combinations, we distinguish between whether (i) the predecessor A_{out} is incident to that face or (ii) not. Figure 4.9b illustrates both configurations, where f_1 denotes the successor located at a face of type (i), and f_2 the successor that is incident to the face at the opposite side (ii). For both subcases, the rerouting rules for the first two edge pairs are relatively simple, since they follow the basic principle of the first case, see Figure 4.11a and 4.11b. However, the third pair is more complicated. For (i) the sequence A_{out}, v_T, f_1 is replaced by A_{out}, C, B, A, f_1, whereas for (ii) A_{out}, v_T, f_1 is substituted by A_{out}, A, B, C, f_2 (Figure 4.11c).

Case 3 (Face $\rightsquigarrow v_T \rightsquigarrow$ Face): Both predecessor and successor of v_T in $H^*_{out}(T)$ are not adjacent to v_T. Hence, the cycle enters and leaves v_T through a face. Again to cover all possibilities, we have to deal with two subcases: (i) the two faces are distinct or (ii) the cycle $H^*_{out}(T)$ leaves through the same face as it enters. Rerouting $H^*_{out}(T)$ in the first subcase (i) works for all three different edge pairs, even without introducing any new face crossings. The three solutions for (i) are displayed in Figure 4.12, where the predecessor is labeled by f_1 and the successor by f_3. So far we have been able to resolve every configuration such that any pair of edges can be selected to be part of $H_{out}(T)$. However, the interesting case is subcase (ii), where the predecessor f_1 and successor f_2 are incident to the same face. While there is a solution for the edge pair $(A, C), (C, B)$ as displayed in Figure 4.12a, the two remaining edge pairs create unresolvable configurations, see Figure 4.12b and 4.12c, respectively. This dilemma is caused by the fact that $H_{out}(T)$ has to either enter or leave T via C. However, C is not accessible from neither f_1 nor f_2 without destroying planarity.

We may summarize the solutions for the different cases as follows: As long as the cycle does not enter and leave v_T via the same face, we can always choose two edges of T in advance and reroute the cycle such that these two edges become part of $H_{out}(T)$. □



(a) G with T and T' (b) Subhamiltonian cycle (c) After rerouting at T

Figure 4.13. (a) Two separating triangles T and T' with vertices $V(T) = \{A, B, C\}$ and $V(T') = \{A', B', C'\}$ and for each two prescribed edges (bold). (b) T and T' replaced by v_T and $v_{T'}$, every non triangular face is stellated by inserting additional vertices (squares) and edges (dashed), and a (sub)hamiltonian cycle H' (bold). (c) Result of applying the corresponding rule to T, creating an unresolvable configuration for T'.

At this point it is tempting to show that we can always find a cycle that avoids crossing a face twice. By using Lemma 4.3, we may obtain such a cycle in a triconnected graph with no separating triangles. This raises the question if we can use it and apply the described rules to obtain a cycle through multiple triangles for which we may specify two edges in advance. We answer this question negatively with a small counterexample.

Consider the triconnected 4-planar graph G shown in Figure 4.13a. It contains two separating triangles T and T' with vertices $V(T) = \{A, B, C\}$ and $V(T') = \{A', B', C'\}$, respectively. In every triangle the two bold drawn edges, that is, $(A', C'), (B', C')$ and $(A, C), (B, C)$, are prescribed to lie on the augmented subhamiltonian cycle H. We proceed as described; both triangles are replaced by a dummy vertex v_T and $v_{T'}$, respectively. The resulting graph (Figure 4.13b) is triconnected 4-planar and free of separating triangles. The squares and dashed lines correspond to the dummy vertices and edges inserted by the technique of Kainen and Overbay [60] as described in Lemma 4.3.

We may now compute a Hamiltonian cycle H' by applying the linear-time algorithm of Chiba and Nishizeki [23]. Assume the result is the bold cycle in Figure 4.13b. Clearly the cycle crosses every face at most once after we remove the dummy vertices inserted by the technique of Kainen and Overbay [60]. We reinsert T and apply the corresponding rule, that is, the augmentation displayed in Figure 4.11b. The result of augmenting such that the two marked edges of T, namely $(A, C), (B, C)$, lie on the cycle is displayed in Figure 4.13c. Notice that we are forced to enter T via A and exit by B. As a result, the cycle crosses one face twice. Moreover, T' must be entered and left through the same face. The corresponding rule, illustrated in Figure 4.12b, implies that

we cannot reroute the cycle such that it contains the edges $(A', C'), (B', C')$. However, we may lift the restriction, use the only rule applicable in this case (Figure 4.12a), and obtain a cycle with edges $(A', C'), (A', B')$ instead. Notice that the graph in this example has even a Hamiltonian cycle H through the requested edges. However, the purpose of the example is to demonstrate that for an arbitrary chosen subhamiltonian cycle, the described rules cannot always be applied. We may conclude that when using Lemma 4.3, we may choose for one (the first) triangle two edges because the initial cycle visits every face at most once. From there on, we can only guarantee that two unknown edges are part of the final cycle. In the following we will benefit from this observation.

Recall the aforementioned single-separating-triangle scenario. Both $\overline{G}_{out}(T)$ and $\overline{G}_{in}(T)$ are free of separating triangles. Thus, we may construct two graphs $G^*_{out}(T), G^*_{in}(T)$ by replacing T with dummy vertices. Applying Lemma 4.3 to them yields two subhamiltonian cycles $H^*_{out}(T)$ and $H^*_{in}(T)$, both crossing every face of $G^*_{out}(T)$ and $G^*_{in}(T)$ at most once. Hence, we may augment them with the aid of Lemma 4.9 such that they contain each two edges of T. By choosing the combination of the edges such that $H_{out}(T)$ and $H_{in}(T)$ meet the requirements of Lemma 4.8, we can merge them into a single subhamiltonian cycle H for G. While the property that $G^*_{out}(T)$ and $G^*_{in}(T)$ are both free of separating triangles enables us to conveniently choose two edges for each cycle $H_{out}(T), H_{in}(T)$, this only works for a single separating triangle. However, a closer look reveals that it is sufficient to have a choice for either $H_{out}(T)$ or $H_{in}(T)$, not necessarily both of them. The idea is to first augment the cycle for which we do not have a choice to see which edges of T are part of it, then we choose the edges for the second cycle accordingly. We summarize the idea as the main result of this section and describe it in a more formal manner in form of a proof.

▶ **Theorem 4.10.** *Every triconnected 4-planar graph is subhamiltonian.*

Proof. Let G denote a triconnected 4-planar graph and $\tau(G)$ the number of separating triangles in G. We prove by induction and claim that for any $\tau(G) \geq 0$, we can compute a subhamiltonian cycle H for G. *Base case*: Since $\tau(G) = 0$, we can directly apply Lemma 4.3. *Inductive case:* For $\tau(G) > 0$, we pick a separating triangle T such that $\tau(\overline{G}_{in}(T)) = 0$. Let $G^*_{out}(T)$ be the result of replacing T by v_T in $\overline{G}_{out}(T)$. Notice that $\tau(G^*_{out}(T)) = \tau(G) - 1$ holds. Hence, by induction hypothesis, $G^*_{out}(T)$ has a subhamiltonian cycle $H^*_{out}(T)$. We reinsert T and augment $H^*_{out}(T)$ such that the result $H_{out}(T)$ contains two (arbitrary) edges e_1, e_2 of T. In a similar way, we replace T in $\overline{G}_{in}(T)$ by v'_T to obtain $G^*_{in}(T)$. Since $\tau(\overline{G}_{in}(T)) = \tau(G^*_{in}(T)) = 0$ holds, we can apply Lemma 4.3 to $G^*_{in}(T)$ and compute a cycle $H^*_{in}(T)$ that crosses each face at most once.

With Lemma 4.9 we may obtain a cycle $H_{in}(T)$ for $\overline{G}_{in}(T)$ with two edges $e'_1, e'_2 \in E(T)$ of our choice. Choosing $e'_1 = e_1$ and $e'_2 \neq e_2$ yields two cycles $H_{out}(T)$ and $H_{in}(T)$ that meet the requirements of Lemma 4.8 and we can merge them into one cycle H for G. □

The proof of Theorem 4.10 is constructive. Embedding G in the plane takes linear time. In order to identify all separating triangles in G within the same time frame, one may apply a naive approach that works as follows. At every vertex v, we test every neighbor of v's neighbors if it is adjacent to v, that is, if v is part of a 3-cycle. If that is the case and this 3-cycle is not a face, which can be tested by examining the edges incident to the vertices of the 3-cycle in the embedding, then a separating triangle has been found. Notice that due to the degree restriction the number of vertices to consider for a fixed v is constant, which yields a linear-time algorithm. Augmenting a subhamiltonian cycle and merging two of them takes constant time. Disjointness of separating triangles yields a linear number of subproblems and every edge occurs in at most one such subproblem. Hence, the total time spent for the subroutine of Lemma 4.3 is linear in the size of G.

▶ **Corollary 4.11.** *A subhamiltonian cycle of a triconnected 4-planar graph can be found in linear time.*

In this section, we have shown that in the triconnected case a rather simple technique can be used to efficiently compute a subhamiltonian cycle in a 4-planar graph. However, the property that G is triconnected has been used extensively throughout this section, thus, a relaxation to biconnectivity is not straightforward. Recall that Kainen and Overbay describe in their work [60] a technique that augments a biconnected planar graph without separating triangles to a triconnected one with the same property. It is tempting to employ this approach here, but the augmentation step may raise the degree of a vertex, leaving us with a graph that is not 4-planar anymore.

Overview of the biconnected 4-planar case

Besides the technique for triconnected 4-planar graphs, we describe in [7, 8] an approach for the general case. The result is a quadratic-time algorithm that embeds every biconnected 4-planar graph into a book with two pages. Recall that one may assume biconnectivity due to Lemma 4.1. Although the biconnected algorithm is not part of this thesis and is quite technical, we give a brief description here. The overall idea follows a principle that has been used by Heath [54] and subsequently by Yannakakis [72] to prove the upper bound of

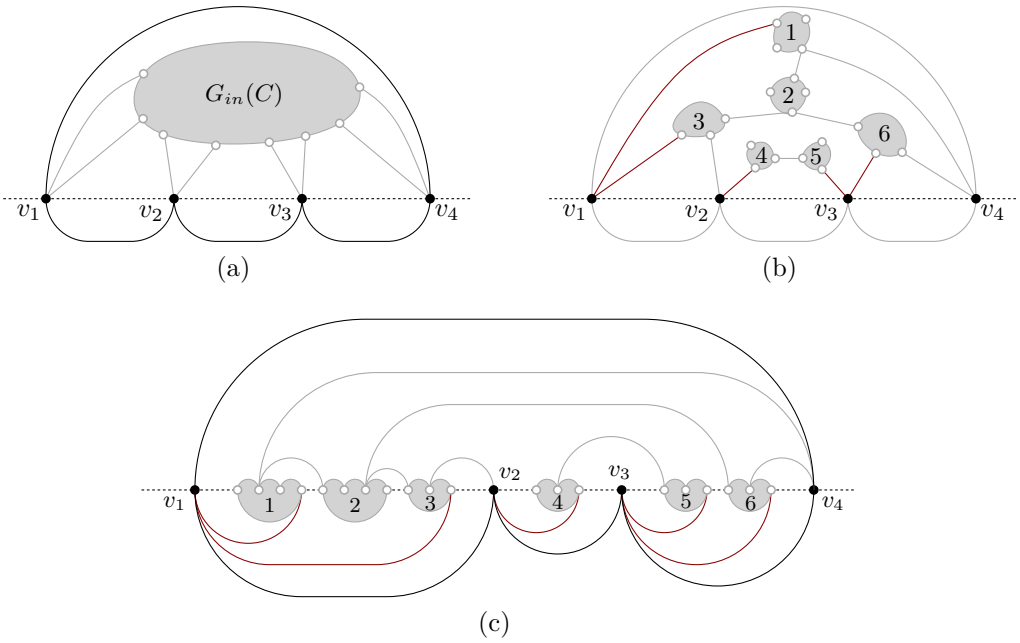


Figure 4.14. (a) Initial layout of the simple chord-less cycle C that bounds $G_{in}(C)$. (b) Bridge-block tree decomposition of $G_{in}(C)$ with red edges being the ones that determine the position of the blocks on the spine. (c) The final placement of all blocks.

seven and four, respectively, for the book thickness of planar graphs. It embeds the graph from the outside to the inside, starting with the outer face, and then in a recursive manner, peeling away cycles. But in difference to general planar graphs, in the 4-planar case, two pages are sufficient.

The intuition behind this approach can best be explained by an example. Suppose we are given some cycle C that, for the beginning, is simple and without chords. Such a cycle is always drawn as illustrated in Figure 4.14a, that is every edge (except one) is assigned to the lower page. The main problem, of course, is everything that is contained inside C. Let this subgraph be $G_{in}(C)$ as shown in Figure 4.14a. $G_{in}(C)$ is decomposed into its bridge-block tree. Notice that in difference to a BC-tree, a block in the bridge-block tree may contain cut vertices. Figure 4.14b shows an example of such a bridge-block tree. The idea is now to treat the blocks as vertices and embedd them on the spine with a proper page assignment for the edges. These edges are of two types: Bridges of the bridge-block tree, and edges that connect C and $G_{in}(C)$. The latter ones determine the position of the blocks on the spine using roughly the following rule: Every block that has at least one neighbor on C is placed directly to the

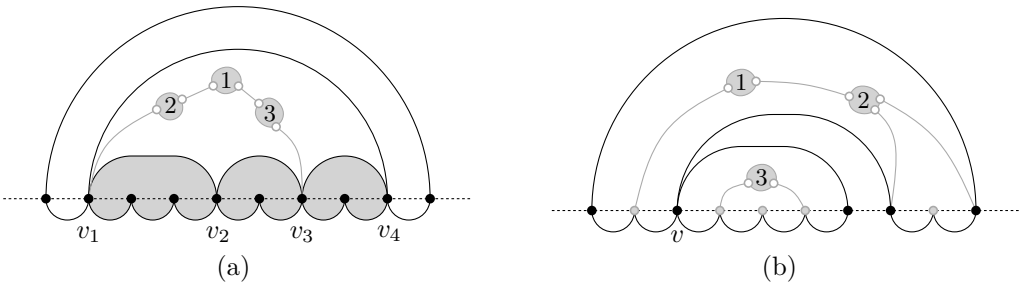


Figure 4.15. (a) Three bridge-blocks trapped inside a cycle of chords $v_1, \ldots, v_4$. (b) A cycle that is not simple with v being a cut vertex in the bridge-block.

right of the leftmost among these neighbors. For those that have no neighbor on C (see for example the second block in Figure 4.14b and 4.14c), one can show [8] that there exists always a suitable spot on the spine that preserves the embedding implied by the original graph.

Suppose now that we have placed the blocks on the spine. Notice that each block itself is bounded by a (not necessarily simple) cycle. Now we draw every cycle that bounds a block as displayed in Figure 4.14c, that is upside down when compared to the layout of C. Applying a recursive argument for these cycles yields a two-page book embedding. One may ask now, why this only works for 4-planar graphs. Recall that we assumed that C is simple and chordless, which of course is not always the case. This is where the degree restriction is important.

Assume that C is a simple cycle that contains chords. These chords cannot be drawn on the lower page, because the embedding cannot be preserved this way. Hence, they have to be drawn together with the other edges on the top page, while at the same time it must be ensured that there is always a gap on the spine to place the blocks. In the general planar case, one may easily construct a scenario such that a couple of blocks are trapped by chords. An example is shown in Figure 4.15a in which two blocks cannot be embedded on the spine due to a cycle of chords surrounding them. This situation is unavoidable in the general planar case, but in the 4-planar case, the degree restriction prevents such a situation. Notice that due to biconnectivity, the trapped blocks must have at least two neighbors on C, both of them exceeding their maximum degree of four. The other problem arises when C is not simple. Recall that C is the outer face of a block and such a block may contain cut vertices. However, a cut vertex is then part of two edge disjoint cycles, thus, having already a degree of four. This makes it possible to nest the cycles, similar to Lemma 4.1. Figure 4.15b shows an example. For details, see Lemma 13 in [8].

4.4 Conclusion

We studied the problem of embedding triconnected 3-planar and 4-planar graphs into a book of two pages. The former result is more of a practical nature, since it does not provide any new theoretical results but yields a simple canonical ordering-based solution to the problem. Thereby lowering considerably the hurdle towards an efficient simple implementation. One question arising naturally: May one employ the bitonic st-ordering instead?

Indeed this seems to be not too difficult. The key component of a possible invariant is to guarantee that a vertex with only one neighbor in G_k has both its predecessor and successor on the subhamiltonian cycle located on the outer face. As a result, the augmentation step can be applied two times, which might be necessary due to the possible two neighbors in $G - G_k$. One may ask now why we did not present this idea in the first place. At this point it is important to keep the big picture in mind. We are given an undirected (biconnected) 3-planar graph and must obtain a bitonic st-ordering. Of course, the previous chapter provides us with such an ordering. However, our initial argument was the simplicity of the canonical ordering-based algorithm compared to the custom face traversal of Heath. Using instead now an algorithm that requires an SPQR-tree decomposition is probably not the right way to go.

However, applying the bitonic st-ordering to the 4-planar case is different. The biconnected 4-planar algorithm is quite involved, therefore, a bitonic st-ordering based algorithm would be of great value. But since one has to deal with the case in which a vertex may have three neighbors in $G - G_k$, finding an invariant, similar to the one used in the 3-planar case, is not straightforward.

Before drawing the final conclusions of this thesis, we would like to state two open problems that arise naturally. The approach for triconnected 4-planar graphs relies heavily on the maximum degree restriction and triconnectivity. It would be interesting to know if a similar approach works by pushing one of the properties a bit further. More precisely, are triconnected 5-planar graphs subhamiltonian and is there a way to circumnavigate the problems arising when extending the presented approach from triconnected to biconnected?

5 Conclusion

In Chapter 3 we have introduced the concept of bitonic *st*-orderings for undirected biconnected planar graphs and planar *st*-graphs. For the former a linear-time algorithm has been described that is based on canonical orderings and an SPQR-tree decomposition. The algorithm is able to provide a bitonic *st*-ordering for every planar biconnected graph at the expense of a possible change in the embedding. Afterwards, an adaptation of the incremental planar straight-line drawing algorithm of de Fraysseix et al. has been described. Motivated by the insight that this algorithm would be able to create an upward planar straight-line drawing in case the bitonic *st*-ordering is an *st*-ordering of a planar *st*-graph, we studied the problem of finding a bitonic *st*-ordering for an embedded planar *st*-graph. Although not every planar *st*-graph admits such an ordering, we were able to fully characterize those that do. Furthermore, a simple linear-time algorithm for recognizing them has been given. Moreover, the algorithm is able to obtain such an ordering if it exists.

However, experiments on random planar *st*-graphs have shown that it is very unlikely that a planar *st*-graph admits a bitonic *st*-ordering. For those for which no such ordering can be found, an alternative solution has been proposed. Based on the idea of splitting specific edges, therefore, introducing additional vertices, a planar *st*-graph can be transformed into one for which a bitonic *st*-ordering exists. We described a linear-time algorithm that finds the smallest set of edges to split. Together with an upper bound on the number of splits, we were able to improve the upper bound of the number of bends required in upward planar poly-line drawings within quadratic area.

Embedding bounded degree graphs into two-page books has been the topic of Chapter 4. We have shown that every triconnected 4-planar graph admits a two-page book embedding. For the 3-planar case, a simple canonical ordering-based algorithm that constructs a subhamiltonian cycle in an incremental manner, has been presented. For the 4-planar case, we have chosen a different approach. After investigating the properties of separating triangles under the degree restriction, we have shown that by decomposing the problem along the separating triangles, one is able to reroute the subhamiltonian cycles of the subproblems such that they can be merged into one subhamiltonian cycle for the original graph. Since the approach solves only the triconnected case, a short overview of the general case has been given.

Bibliography

[1] S. Abbasi, P. Healy, and A. Rextin. Improving the running time of embedded upward planarity testing. *Information Processing Letters*, 110(7):274–278, 2010.

[2] M. J. Alam, T. C. Biedl, S. Felsner, A. Gerasch, M. Kaufmann, and S. Kobourov. Linear-time algorithms for hole-free rectilinear proportional contact graph representations. In T. Asano, S.-i. Nakano, Y. Okamoto, and O. Watanabe, editors, *Algorithms and Computation*, volume 7074 of *Lecture Notes in Computer Science*, pages 281–291. Springer, 2011.

[3] M. J. Alam, T. C. Biedl, S. Felsner, M. Kaufmann, S. G. Kobourov, and T. Ueckerdt. Computing cartograms with optimal complexity. In *Proceedings of the Twenty-eighth Annual Symposium on Computational Geometry*, SoCG '12, pages 21–30. ACM, 2012.

[4] M. Badent, U. Brandes, and S. Cornelsen. More canonical ordering. *Journal of Graph Algorithms and Applications*, 15(1):97–126, 2011.

[5] M. A. Bekos, M. Gronemann, M. Kaufmann, and R. Krug. Planar octilinear drawings with one bend per edge. In C. Duncan and A. Symvonis, editors, *Graph Drawing*, volume 8871 of *Lecture Notes in Computer Science*, pages 331–342. Springer Berlin Heidelberg, 2014.

[6] M. A. Bekos, M. Gronemann, S. Pupyrev, and C. Raftopoulou. Perfect smooth orthogonal drawings. In *The 5th International Conference on Information, Intelligence, Systems and Applications, IISA 2014*, pages 76–81, July 2014.

[7] M. A. Bekos, M. Gronemann, and C. N. Raftopoulou. Two-page book embeddings of 4-planar graphs. In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, pages 137–148, 2014.

[8] M. A. Bekos, M. Gronemann, and C. N. Raftopoulou. Two-page book embeddings of 4-planar graphs. *CoRR*, abs/1401.0684, 2014.

[9] M. A. Bekos, M. Kaufmann, S. Kobourov, and A. Symvonis. Smooth orthogonal layouts. In W. Didimo and M. Patrignani, editors, *Graph Drawing*, volume 7704 of *Lecture Notes in Computer Science*, pages 150–161. Springer Berlin Heidelberg, 2013.

[10] F. Bernhart and P. C. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory*, 27(3):320–331, 1979.

[11] P. Bertolazzi, G. Di Battista, and W. Didimo. Quasi-upward planarity. In S. Whitesides, editor, *Graph Drawing*, volume 1547 of *Lecture Notes in Computer Science*, pages 15–29. Springer Berlin Heidelberg, 1998.

[12] P. Bertolazzi, G. Di Battista, G. Liotta, and C. Mannino. Upward drawings of triconnected digraphs. *Algorithmica*, 12(6):476–497, 1994.

[13] P. Bertolazzi, G. Di Battista, C. Mannino, and R. Tamassia. Optimal upward planarity testing of single-source digraphs. In T. Lengauer, editor, *Algorithms—ESA '93*, volume 726 of *Lecture Notes in Computer Science*, pages 37–48. Springer Berlin Heidelberg, 1993.

[14] T. C. Biedl. Transforming planar graph drawings while maintaining height. *CoRR*, abs/1308.6693, 2013.

[15] T. C. Biedl. Height-preserving transformations of planar graph drawings. In C. Duncan and A. Symvonis, editors, *Graph Drawing*, volume 8871 of *Lecture Notes in Computer Science*, pages 380–391. Springer Berlin Heidelberg, 2014.

[16] T. C. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. In J. van Leeuwen, editor, *Algorithms — ESA '94*, volume 855 of *Lecture Notes in Computer Science*, pages 24–35. Springer Berlin Heidelberg, 1994.

[17] G. Blin, G. Fertin, I. Rusu, and C. Sinoquet. Extending the hardness of rna secondary structure comparison. In B. Chen, M. Paterson, and G. Zhang, editors, *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, volume 4614 of *Lecture Notes in Computer Science*, pages 140–151. Springer Berlin Heidelberg, 2007.

[18] J. M. Boyer and W. J. Myrvold. On the cutting edge: Simplified o(n) planarity by edge addition. *Journal of Graph Algorithms and Applications*, 8(3):241–273, 2004.

[19] F. J. Brandenburg. Drawing planar graphs on area $\frac{4}{3}n \times \frac{2}{3}n$. *Electronic Notes in Discrete Mathematics*, 31(0):37 – 40, 2008. The International Conference on Topological and Geometric Graph Theory.

[20] U. Brandes. Eager st-ordering. In R. Möhring and R. Raman, editors, *Algorithms — ESA 2002*, volume 2461 of *Lecture Notes in Computer Science*, pages 247–256. Springer Berlin Heidelberg, 2002.

[21] C. Chen. Any maximal planar graph with only one separating triangle is hamiltonian. *Journal of Combinatorial Optimization*, 7(1):79–86, 2003.

[22] C. C. Cheng, C. A. Duncan, M. T. Goodrich, and S. G. Kobourov. Drawing planar graphs with circular arcs. *Discrete and Computational Geometry*, 25:405–418, 2001.

[23] N. Chiba and T. Nishizeki. The hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. *Journal of Algorithms*, 10(2):187–211, 1989.

[24] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using PQ-trees. *J. Comput. Syst. Sci.*, 30(1):54–76, 1985.

[25] M. Chimani and R. Zeranski. Upward planarity testing: A computational study. In S. Wismath and A. Wolff, editors, *Graph Drawing*, volume 8242 of *Lecture Notes in Computer Science*, pages 13–24. Springer International Publishing, 2013.

[26] M. Chimani and R. Zeranski. Upward planarity testing via sat. In W. Didimo and M. Patrignani, editors, *Graph Drawing*, volume 7704 of *Lecture Notes in Computer Science*, pages 248–259. Springer Berlin Heidelberg, 2013.

[27] M. Chrobak and G. Kant. Convex grid drawings of 3-connected planar graphs. *Intl. Journal of Computational Geometry and Applications*, 7(3):211–223, 1997.

[28] M. Chrobak and S.-i. Nakano. Minimum-width grid drawings of plane graphs. *Computational Geometry*, 11(1):29–54, 8 1998.

[29] M. Chrobak and T. Payne. A linear-time algorithm for drawing a planar graph on a grid. *Information Processing Letters*, 54(4):241 – 246, 1995.

[30] F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg. Embedding graphs in books: A layout problem with applications to VLSI design. *SIAM Journal on Algebraic and Discrete Methods*, 8(1):33–58, 1987.

[31] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[32] H. de Fraysseix, P. O. de Mendez, and P. Rosenstiehl. Bipolar orientations revisited. *Discrete Applied Mathematics*, 56(2–3):157 – 179, 1995. Fifth Franco-Japanese Days.

[33] H. de Fraysseix, J. Pach, and R. Pollack. Small sets supporting Fary embeddings of planar graphs. In *Procs. 20th Symposium on Theory of Computing (STOC)*, pages 426–433, 1988.

[34] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[35] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Englewood Cliffs, NJ, 1999.

[36] G. Di Battista and F. Frati. A survey on small-area planar graph drawing. *ArXiv e-prints*, Oct. 2014.

[37] G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61(2–3):175 – 198, 1988.

[38] G. Di Battista and R. Tamassia. Incremental planarity testing. In *30th Annual Symposium on Foundations of Computer Science, 1989*, pages 436–441, 1989.

[39] G. Di Battista, R. Tamassia, and I. Tollis. Area requirement and symmetry display of planar upward drawings. *Discrete & Computational Geometry*, 7(1):381–401, 1992.

[40] E. Di Giacomo, W. Didimo, M. Kaufmann, G. Liotta, and F. Montecchiani. Upward-rightward planar drawings. In *The 5th International Conference on Information, Intelligence, Systems and Applications, IISA 2014*, pages 145–150, July 2014.

[41] E. Di Giacomo and G. Liotta. The hamiltonian augmentation problem and its applications to graph drawing. In M. Rahman and S. Fujita, editors, *WALCOM: Algorithms and Computation*, volume 5942 of *Lecture Notes in Computer Science*, pages 35–46. Springer Berlin Heidelberg, 2010.

[42] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer Berlin Heidelberg, 3rd edition.

[43] V. Dujmovic and D. R. Wood. On linear layouts of graphs. *Discrete Mathematics & Theoretical Computer Science*, 6(2):339–358, 2004.

[44] C. Duncan, E. Gansner, Y. Hu, M. Kaufmann, and S. Kobourov. Optimal polygonal representation of planar graphs. *Algorithmica*, 63(3):672–691, 2012.

[45] C. Duncan and S. Kobourov. Polar coordinate drawing of planar graphs with good angular resolution. *Journal of Graph Algorithms and Applications*, 7(4):311–333, 2003.

[46] J. Ebert. st-ordering the vertices of biconnected graphs. *Computing*, 30(1):19–33, 1983.

[47] S. Even and R. E. Tarjan. Computing an st-numbering. *Theoretical Computer Science*, 2(3):339 – 344, 1976.

[48] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[49] A. Garg and R. Tamassia. On the computational complexity of upward and rectilinear planarity testing. In R. Tamassia and I. Tollis, editors, *Graph Drawing*, volume 894 of *Lecture Notes in Computer Science*, pages 286–297. Springer Berlin Heidelberg, 1995.

[50] M. Gronemann. Bitonic st-orderings of biconnected planar graphs. In C. Duncan and A. Symvonis, editors, *Graph Drawing*, volume 8871 of *Lecture Notes in Computer Science*, pages 162–173. Springer Berlin Heidelberg, 2014.

[51] C. Gutwenger and P. Mutzel. Planar polyline drawings with good angular resolution. In S. Whitesides, editor, *Graph Drawing*, volume 1547 of *Lecture Notes in Computer Science*, pages 167–182. Springer Berlin Heidelberg, 1998.

[52] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In J. Marks, editor, *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer Berlin Heidelberg, 2001.

[53] D. Harel and M. Sardas. An algorithm for straight-line drawing of planar graphs. *Algorithmica*, 20(2):119–135, 1998.

[54] L. S. Heath. Embedding planar graphs in seven pages. In *Foundations of Computer Science (FOCS '84)*, pages 74–83. IEEE, 1984.

[55] L. S. Heath. *Algorithms for Embedding Graphs in Books*. PhD thesis, University of North Carolina, Chapel Hill, 1985.

[56] G. Helden. Each maximal planar graph with exactly two separating triangles is hamiltonian. *Discrete Applied Mathematics*, 155(14):1833–1836, 2007.

[57] G. Helden. *Hamiltonicity of maximal planar graphs and planar triangulations*. PhD thesis, RWTH Aachen, 2007.

[58] J. Hopcroft and R. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, Oct. 1974.

[59] M. Jünger and P. Mutzel, editors. *Graph Drawing Software*. Springer Berlin Heidelberg, 2004.

[60] P. C. Kainen and S. Overbay. Extension of a theorem of Whitney. *Applied Mathematics Letters*, 20(7):835–837, 2007.

[61] G. Kant. Hexagonal grid drawings. In *18th Workshop on Graph-Theoretic Concepts in Computer Science*, pages 263–276, 1992.

[62] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996.

[63] K. Mehlhorn and P. Mutzel. On the embedding phase of the hopcroft and tarjan planarity testing algorithm. *Algorithmica*, 16(2):233–242, 1996.

[64] K. Menger. Zur allgemeinen kurventheorie. *Fund. Math.*, 10:96–115, 1927.

[65] OGDF - Open Graph Drawing Framework. http://www.ogdf.net/.

[66] R. Tamassia. *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2007.

[67] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[68] W. T. Tutte. A theorem on planar graphs. *Transactions of the American Mathematical Society*, 82(1):99–116, 1956.

[69] H. Whitney. A theorem on graphs. *Annals of Mathematics*, 32(2):378–390, 1931.

[70] A. Wigderson. The complexity of the hamiltonian circuit problem for maximal planar graphs. Technical Report TR-298, EECS Department, Princeton University, 1982.

[71] M. Yannakakis. Four pages are necessary and sufficient for planar graphs. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 104–108, New York, NY, USA, 1986. ACM.

[72] M. Yannakakis. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38(1):36–67, 1989.

[73] K.-H. Yeap and M. Sarrafzadeh. Floor-planning by graph dualization: 2-concave rectilinear modules. *SIAM Journal on Computing*, 22(3):500–526, 1993.

Erklärung

Ich versichere, dass ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; dass diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie – abgesehen von unten angegebenen Teilpublikationen – noch nicht veröffentlicht worden ist sowie, dass ich eine solche Veröffentlichung vor Abschluss des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen der Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. Michael Jünger betreut worden.

Köln, 27. April 2015

M. Gronemann

Teilpublikationen

- M. Gronemann. Bitonic st-orderings of biconnected planar graphs. In C. Duncan and A. Symvonis, editors, *Graph Drawing*, volume 8871 of *Lecture Notes in Computer Science*, pages 162–173. Springer Berlin Heidelberg, 2014.
- M. A. Bekos, M. Gronemann, and C. N. Raftopoulou. Two-page book embeddings of 4-planar graphs. In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, pages 137–148, 2014.
- M. A. Bekos, M. Gronemann, and C. N. Raftopoulou. Two-page book embeddings of 4-planar graphs. *CoRR*, abs/1401.0684, 2014.