Karlsruhe Institute of Technology

On Interval Planar Graphs

Bachelor Thesis of

Paul Jungeblut

At the Department of Informatics
Institute of Theoretical Informatics

Reviewers:	Prof. Dr. Dorothea Wagner
	Prof. Dr. Peter Sanders
Advisors:	Lukas Barth
	Guido Brückner
	Marcel Radermacher

Time Period: 22 June 2017 – 21 October 2017

Statement of Authorship

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text.

Karlsruhe, September 29, 2017

Abstract

A planar drawing of a directed graph is upward planar, if each edge is a y-monotone curve. The problem to find such drawings is known as Upward Planarity. In Level Planarity we ask for an upward planar drawing where each vertex has a fixed y-coordinate previously assigned to it. We will close the gap between Upward Planarity and Level Planarity by introducing a new, more general problem that we call Interval Planarity. In this problem we ask for an upward planar drawing of a directed graph $G = (V, E)$, where each vertex has its y-coordinate chosen from an interval that was previously assigned to it.
We show NP-completeness of the problem in its general form and consider several restricted variants that allow an efficient upward planarity test, since this is a necessary condition. We describe embedding algorithms for st-planar graphs and single source graphs with a fixed combinatorial embedding in $\mathcal{O}(|V|)$ time and cycles in $\mathcal{O}(|V|^5)$ time. On the other hand we show that the problem remains NP-complete, even if a combinatorial embedding is fixed or the graph is restricted to be a tree, single source or outerplanar.
We finish by showing that Interval Planarity is also NP-complete when we bound the maximum number of vertices with the same y-coordinate, even for st-planar graphs or very simple trees.

Deutsche Zusammenfassung

Eine planare Zeichnung eines Graphen ist aufwärtsplanar, wenn jede Kante durch eine y-monotone Kurve dargestellt wird. Das Problem, eine solche Zeichnung zu finden, heißt Upward Planarity. Bei Level Planarity geht es darum, eine solche aufwärtsplanare Zeichnung zu finden, in der jeder Knoten eine vorher festgelegte y-Koordinate hat. Wir werden die Lücke zwischen Upward Planarity und Level Planarity schließen, indem wir ein neues, allgemeineres Problem namens Interval Planarity einführen. Darin wird eine aufwärtsplanare Zeichnung gesucht, bei der jeder Knoten seine y-Koordinate aus einem ihm zugewiesenem Intervall haben muss. Wir zeigen die NP-Vollständigkeit des Problems in seiner allgemeinsten Form und werden uns dann mit Spezialfällen beschäftigen, für die Upward Planarity als notwendige Voraussetzung effizient überprüft werden kann. Wir werden Einbettungsalgorithmen mit Laufzeit in $\mathcal{O}(|V|)$ für st-planare Graphen und Graphen mit nur einer Quelle und einer festen kombinatorischen Einbettung beschreiben. Für Kreise werden wir einen Algorithmus mit Laufzeit in $\mathcal{O}(|V|^5)$ angeben. Darüberhinaus zeigen wir die NP-Vollständigkeit von Instanzen mit fester kombinatorischer Einbettung. Ist der zugrundeliegende Graph ein Baum, außenplanar oder hat nur eine einzige Quelle, zeigen wir ebenfalls die NP-Vollständigkeit.
Abschließend betrachten wir die Variante des Problems, in der die maximale Anzahl an Knoten mit der gleichen y-Koordinate beschränkt ist. Interval Planarity ist auch in diesem Fall NP-vollständig, selbst für st-planare Graphen und sehr einfache Bäume.

Contents

1. Introduction

In an upward planar drawing of a directed graph $G = (V, E)$, each edge is drawn as a y-monotone upward curve. The problem to find such a drawing is known as Upward Planarity. Upward planar drawings are commonly used to display hierarchical networks that appear in many areas. Consider a class diagram with inheritance as a software engineering example or a Hasse Diagram of a partially ordered set. In some applications the hierarchy further implies a partition of the objects into several levels. These objects should be on the same rank in the hierarchy. Think of subclasses of the same superclass as an example. In this case we talk about level planar drawings of a graph G and the problem to find these drawings is known as Level Planarity. Obviously Level Planarity is a special case of Upward Planarity and as it turns out it is always testable in linear time, while Upward Planarity is NP-complete. In this thesis we want to close the gap between the two. We will do this by restricting the possible y-coordinates of each vertex. Not necessarily to only one integer number as in Level Planarity but to some interval of integers, so that we get an adjustable degree of freedom for each vertex. We will call this problem as Interval Planarity.

Lots of research has been done regarding Upward Planarity and Level Planarity. Garg et al. proved the NP-completeness of Upward Planarity [8], while Jünger and Leipert showed that a level planar embedding can be constructed in linear time if it exists [11]. Efficient upward planarity tests have been described for many restricted graph classes, for example single source graphs [1, 10], st-planar graphs [4], trees [4] and outerplanar graphs [13]. Throughout this thesis we will consider those and see how the computational complexity changes when the restrictions induced by the intervals are applied.

Chapter 2 will cover the necessary prerequisites. We will formally define Upward Planarity, Level Planarity and Interval Planarity, the new problem we introduce and explore in this thesis. Further we will describe some tools that we will later use to develop and proof graph embedding algorithms.

The existence of interval planar embeddings is considered in Chapter 3. We start by taking a look at the intervals assigned to each vertex in Section 3.1. In some cases we can see that no interval planar embedding exists just by looking at the intervals of the two endpoints of each edge. As a first graph class we consider st-planar graphs in Section 3.2 and we will give a linear time algorithm to find polyline or straight line embeddings. Using this result we will give a combinatorial characterization of interval planar graphs in Section 3.3. This will

lead us to a general embedding strategy to find interval planar embeddings: First we try to augment the graph into an interval planar st-planar graph and then apply the methods for st-planar graphs. The remainder of the chapter is spent considering several other special graph classes. We show embedding algorithms for single source graphs with a fixed combinatorial embedding (Section 3.5.1) and cycles (Section 3.7). Additionally we show NP-completeness for general single source graphs (Section 3.5.2), trees and outerplanar graphs (Section 3.6) and graphs with a fixed combinatorial embedding. All of these are efficiently testable for upward planarity, but not for interval planarity.

Chapter 4 shows the similarity between interval planarity and multiprocessor scheduling problems. Finding a schedule for a set of tasks with unit processing times, release times, deadlines and precedences between the tasks corresponds to an instance of Interval Planarity except for the missing planarity constraint. We will use known scheduling methods to assign levels to the vertices and will then describe how this level assignment can be used to create interval planar embeddings in Section 4.2. By using methods to find schedules on a minimum number of machines we will show that it is NP-complete to find an interval planar embedding of a graph with a minimal number of vertices sharing the same y-coordinate in Section 4.3.

2. Preliminaries

We will use this chapter to formally define Upward Planarity and Level Planarity, the two extreme cases of Interval Planarity, which will be introduced and explored throughout this thesis. Additionally describe the *ear-decomposition*, a common way to decompose graphs. It will be used in Chapter 3 to develop graph embedding algorithms. At last we will summarize some basic properties of Bézier curves, that we will use to construct straight line drawings in Section 3.2.

2.1. Upward, Level and Interval Planarity

This section looks at different planar embeddings for a directed graph. When we talk about embeddings we can further distinguish between so called *topological embeddings* and *combinatorial embeddings*. A topological embedding fixes the exact coordinates of each vertex and the exact plane curve for each edge between two vertices. A combinatorial embedding allows much more freedom: For every vertex it only fixes the cyclic order of the edges incident to it. This already uniquely defines the faces of the graph. However as shown in Figure 2.1, it does not force any face to be the *outer face*, which is the only unbounded face in any topological embedding. Actually any face can be the outer face. We will see later, that fixing a combinatorial embedding and the outer face sometimes leads to efficient embedding algorithms for instances that would otherwise be NP-hard.

We say that a topological embedding of a graph G is *planar*, if no two edges intersect, except maybe at a common endpoint. If such an embedding exists we call G planar. A topological embedding is *upward*, if all edges are y-monotone curves strictly increasing from their start- to their end-vertex. In this case, G is called upward.

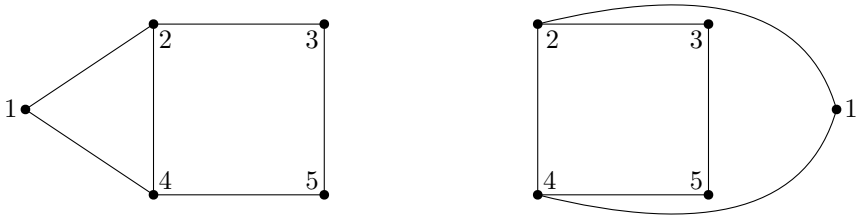

Figure 2.1.: The same graph on the left and on the right side with the same combinatorial embedding (cyclic order of the edges around each vertex). On the left, face $\{1, 2, 3, 4, 5\}$ is the outer face. On the right, face $\{1, 2, 4\}$ is the outer face.

Definition 2.1 Upward Planarity

Instance Given a directed acyclic graph $G = (V, E)$.

Question Is there a topological embedding of G that is upward and planar? If so, such an embedding is called *upward planar*.

In an upward planar embedding the y-coordinates of the vertices can be arbitrary real numbers. Level Planarity simplifies the problem by already setting the y-coordinate of each vertex to an integer, thus implicitly partitioning the vertex set into *levels* containing vertices with the same y-coordinate. The horizontal order of the vertices in each level can still be chosen arbitrarily.

Definition 2.2 Level Planarity

Instance Given a directed acyclic graph $G = (V, E)$ and a leveling function $l : V \to \mathbb{N}$.

Question Is there a topological embedding of G that is planar and each vertex $v \in V$ has y-coordinate $l(v)$? If so, such an embedding is called *level planar*.

Both problems are well studied and understood. Upward Planarity is NP-complete in the general case as shown by Garg and Tamassia [8], but several graph classes allow efficient upward planarity testing. We will systematically study them in Chapter 3. Level Planarity can always be tested in linear time as shown by Jünger and Leipert [11].

Definition 2.3 Interval Planarity

Instance Given a directed acyclic graph $G = (V, E)$ and a function I, that maps each vertex $v \in V$ to an interval $I(v) \subseteq \mathbb{Z}$ of integers.

Question Is there an upward planar embedding of G where each vertex $v \in V$ has its y-coordinate contained in its interval $I(v)$? If so, such an embedding is called *interval planar*.

The definition of Interval Planarity is motivated by the similarity of Upward Planarity and Level Planarity. The goal was to extract their commonalities and to treat them as variants of this more general problem. We observe that Upward Planarity and Level Planarity are indeed special cases of Interval Planarity. For Upward Planarity we choose $I(v) = \mathbb{Z}$ for all $v \in V$. Restricting the y-coordinates to be integers instead of real numbers is justified by the fact that an upward planar embedding of some graph $G = (V, E)$ exists if and only if an interval planar embedding of G exists with $I(v) = \mathbb{Z}$ for all $v \in V$. This is, because only a finite number of distinct real y-coordinates can be chosen for the vertices. Those can then be injectively mapped to the integers $\mathbb{Z}$. For Level Planarity the interval of each vertex consists of a single integer.

We summarize this section in Figure 2.2. It shows the same graph in three different variants. In (a) the vertices are not annotated and the given embedding is upward planar. In (b) the vertices have been annotated with levels. In this particular instance no level planar embedding exists, because vertices d and e need to be at the same level. This can be fixed as in (c) by allowing two different levels for vertices d and e. Note that this also allows to change their relative order: In the given interval planar embedding now e lies below d.

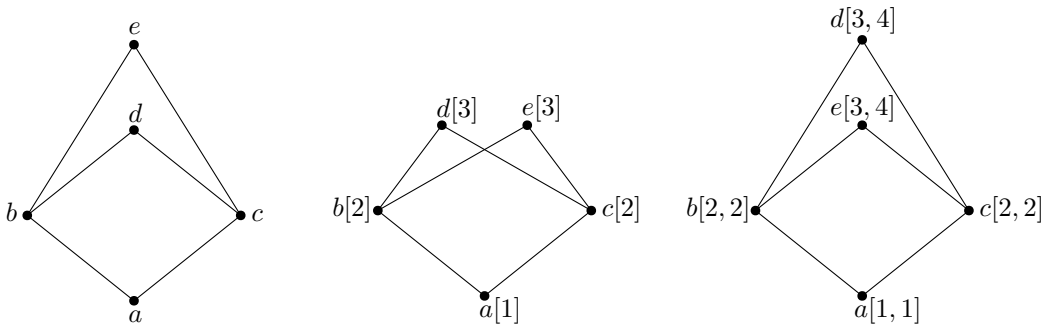

Figure 2.2.: (a) A graph and an upward planar embedding. (b) The same graph with levels for each vertex. This graph does not have a level planar embedding under the given level assignment. (c) The same graph, now with intervals on each vertex. An interval planar embedding under this interval assignment is given.

2.2. Ear-Decomposition

An ear-decomposition is a standard method to partition an undirected and biconnected graph into a cycle and several paths (called ears) that can be glued together to obtain the original graph. We will now describe a very similar decomposition for directed graphs, which we will also call ear-decomposition. We are going to use it in Section 3.2 to embed st-planar graphs.

Classical ear-decomposition is only defined for biconnected graphs and we need to make a similar restriction for directed graphs. We are only considering so called st-planar graphs, whose underlying undirected graph is indeed biconnected.

Definition 2.4 An *st-planar graph* $G = (V, E)$ is an embedded planar directed acyclic graph with exactly one source $s \in V$ and one sink $t \in V$, that both lie on the boundary of the outer face.

Definition 2.5 An *ear* is a directed simple path. An *ear-decomposition* of an st-planar graph $G = (V, E)$ is a partition of the edge set E into a sequence of ears $E_1, \ldots, E_k$. For each but the first ear we require the following properties:

- The two endpoints of the ear belong to earlier ears in the sequence.
- No internal vertices of the ear belong to any earlier ear.

Lemma 2.6 *Let $G = (V, E)$ be an st-planar graph and Γ be an upward planar embedding of G. Then for all vertices $v \in V$, a total left-to-right order of the outgoing edges of v is fixed by Γ.*

Proof Let $v \in V$ be an arbitrary vertex of G. Γ fixes a cyclic order of all outgoing edges of v. Further, all outgoing edges leave v into the half plane above v, because Γ is upward. This then defines a left-to-right order of the outgoing edges of v: Just order them by their clockwise order in Γ. ∎

Assume a graph $G = (V, E)$ is given with an upward planar embedding Γ and an ear decomposition $E_1, \ldots, E_k$. Further let s and t be the first and last vertex of E_1 respectively. The sequence of ears induces a set of subgraphs $G_i := \bigcup_{j=1}^{i} E_j$ for each i $(1 \leq i \leq k)$. Each $G_i \subseteq G$ can be embedded according to Γ. We define the *right front* of G_i to be the rightmost path from s to t in G_i, which always exists, because $s, t \in E_1$. A path P

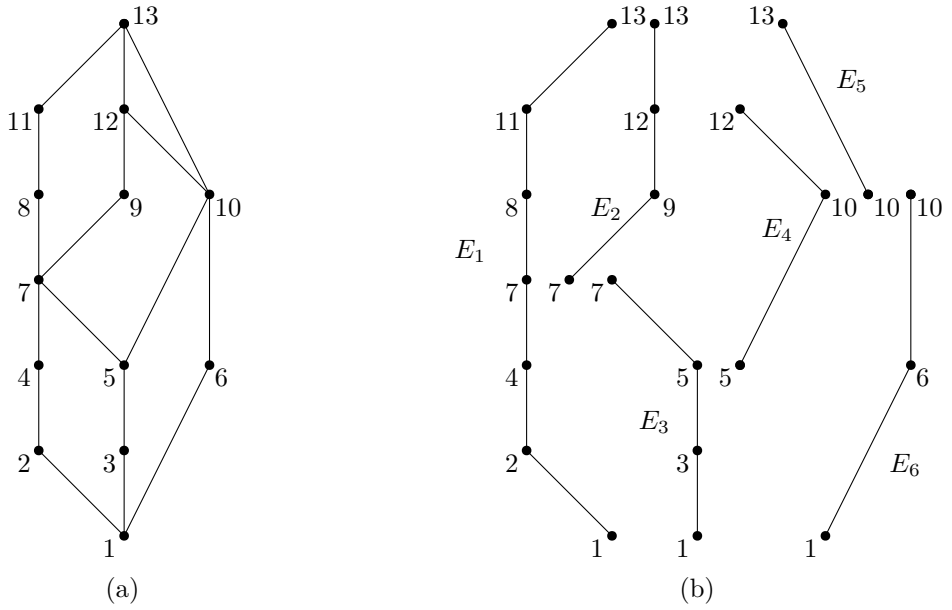

Figure 2.3.: (a) An st-planar graph ($s = 1$, $t = 13$). (b) An ear-decomposition of the graph on the left. Vertices labeled the same correspond to the same vertex in the original graph. $E_1 = (1 \to 2 \to 4 \to 7 \to 8 \to 11 \to 13)$, $E_2 = (7 \to 9 \to 12 \to 13)$, $E_3 = (1 \to 3 \to 5 \to 7)$, $E_4 = (5 \to 10 \to 12)$, $E_5 = (10 \to 13)$, $E_6 = (1 \to 6 \to 10)$

is *rightmost* in G_i, if there is no other st-path in G_i containing an edge that lies right of an edge of P. The rightmost path is well defined, since Lemma 2.6 gave us the total left-to-right order of all outgoing edges at each vertex.

Definition 2.7 An ear-decomposition $E_1, \ldots, E_k$ of an st-planar graph $G = (V, E)$ is called *left-to-right*, if s and t are the endpoints of E_1 and for each but the first ear E_i $(2 \leq i \leq k)$ the following two conditions hold:

- The two endpoints are on the right front of G_{i-1}.
- E_i lies completely on the right front of G_i.

An example graph with a left-to-right ear-decomposition is given in Figure 2.3. The left-to-right order will allow us to embed st-planar graphs ear by ear and we can then use induction on the number of ears to show the correctness of this approach. What is left to show is that such an ear-decomposition exists for all st-planar graphs.

Theorem 2.8 *For every st-planar graph G a left-to-right ear decomposition exists.*

Proof Consider any upward planar embedding Γ of G. We want to construct a sequence $E_1, \ldots, E_k$ of ears for a left-to-right ear-decomposition.

Choose E_1 to be the leftmost path from s to t in Γ. By Lemma 2.6 we know that E_1 is well defined. With each new ear we will extend the current subgraph $G_i = \bigcup_{j=1}^{i} E_j$ until $G_k = G$. At each step i $(2 \leq i \leq k)$ we call an edge $e \in E$ *visited*, if $e \in E(G_{i-1})$. Otherwise it is *unvisited*. Choose E_i to be the leftmost path of unvisited edges between two vertices u and v on the right front of G_{i-1}, where u and v are chosen as follows:

- u is the vertex closest to t that still has unvisited outgoing edges.
- v is the first vertex of G_{i-1} that we reach following the leftmost path of unvisited edges starting at u.

By construction each ear E_i $(2 \leq i \leq k)$ chosen like this intersects G_{i-1} at exactly two vertices, so we get a valid ear-decomposition. Further, ear E_i will be part of the right front of G_i, because otherwise a previous ear was not chosen leftmost, so the ear-decomposition is left-to-right. ∎

The proof of Theorem 2.8 gave a construction for a left-to-right ear-decomposition. Let us again consider Figure 2.3 as an example. The given ear-decomposition was created via this construction. $E_1 = (1 \to 2 \to 4 \to 7 \to 8 \to 11 \to 13)$ is the leftmost st-path. For E_2, vertex 7 is the highest vertex on E_1 that has unvisited outgoing edges and $E_2 = (7 \to 9 \to 12 \to 13)$ is the leftmost path of unvisited edges starting there. Vertex 1 is then the only vertex with unvisited outgoing edges in $G_2 = E_1 \cup E_2$, so $E_3 = (1 \to 3 \to 5 \to 7)$ started at vertex 1. Continuing like this yields the given left-to-right ear-decomposition.

Lemma 2.9 *A left-to-right ear-decomposition of an st-planar graph $G = (V, E)$ can be constructed in $\mathcal{O}(|V|)$ time.*

Proof The proof of Theorem 2.8 gave a construction for a left-to-right ear-decomposition. This can be implemented in $\mathcal{O}(|V|)$ with a left-first depth first search (DFS) from source vertex s. The DFS partitions the edges E into tree-edges and non-tree-edges. Now number the vertices in increasing order by their *finishing time*, that is when all their children are visited and the DFS backtracks to their parent.

The left-to-right ear-decomposition can now be obtained as follows: E_1 is the unique path of tree-edges from s to t. By construction, it is the leftmost path. Mark these edges as visited. For each other ear E_i: Visit the vertices in ascending order of their finishing times: If the current vertex u has unvisited outgoing edges, follow the leftmost unvisited tree-edges as far as possible. We arrive at some vertex v where all outgoing tree-edges are already visited. Then this path and the leftmost unvisited non-tree edge are the next ear E_i. Mark these edges as visited. By construction E_i intersects the right front of the union of G_{i-1} only in its endpoints. It also lies on the right of G_i, because of the left-first order in the DFS, so the ears are left-to-right.

The left-first DFS needs time in $\mathcal{O}(|V|)$. Constructing the ears from the DFS-tree then needs time $\mathcal{O}(|V(E_i)|)$ for ear E_i and thus $\mathcal{O}(|V|)$ in total. ∎

2.3. Bézier Curves

To finish this chapter we are going to look at some properties of Bézier curves that we are going to use to find straight line embeddings of st-planar graphs in Section 3.2.3.

Definition 2.10 The polynomial

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

is called the *i-th Bernstein polynomial of order n*. In this definition we assume that $0^0 := 1$. For $t \in [0, 1]$ a *Bézier curve* for control points $P_0, \ldots, P_n$ is defined as

$$C(t) = \sum_{i=0}^{n} B_{i,n}(t) \cdot P_i.$$

Lemma 2.11 *For $t \in [0, 1]$, the Bézier curve $C(t)$ lies inside the convex hull of its control polygon $P_0, \ldots, P_n$.*

PROOF We just need to show that the Bernstein polynomials of order n are a partition of unity for $t \in [0,1]$:

$$\sum_{i=0}^{n} B_{i,n}(t) = \sum_{i=0}^{n} \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i} = (t + (1-t))^n = 1^n = 1$$

■

Lemma 2.12 *For $t \in [0,1]$ the Bézier curve $C(t)$ interpolates exactly the control points P_0 and P_n and no others.*

PROOF For $t = 0$ and $t = 1$ we get

$$C(0) = \sum_{i=0}^{n} B_{i,n}(0) \cdot P_i = \sum_{i=0}^{n} \binom{n}{i} \cdot \underbrace{0^i \cdot 1^{n-i}}_{0 \text{ for } i \neq 0} \cdot P_i = P_0 \quad \text{and}$$

$$C(1) = \sum_{i=0}^{n} B_{i,n}(1) \cdot P_i = \sum_{i=0}^{n} \binom{n}{i} \cdot \underbrace{1^i \cdot 0^{n-i}}_{0 \text{ for } i \neq n} \cdot P_i = P_n$$

so P_0 and P_n are interpolated. Remember that we assumed $0^0 := 1$. For every $t \in (0,1)$ the term $t^i \cdot (1-t)^{n-i}$ is in $(0,1)$ as well, so every other point on C in influenced by all control points and C does not intersect any of them. ■

Lemma 2.13 *The tangents in the endpoints of the Bézier curve $C(t)$ are parallel to the line through the first two and last two control points respectively.*

PROOF $C(t)$ is continuous in the interval $[0,1]$ and differentiable in $(0,1)$ as a composition of continuous and differentiable functions. With the product and the chain rule we get:

$$C'(t) = \sum_{i=0}^{n} \binom{n}{i} \cdot \left(i \cdot t^{i-1} \cdot (1-t)^{n-i} - t^i \cdot (n-i) \cdot (1-t)^{n-i-1} \right) \cdot P_i$$

To get the tangents at $t = 0$ and $t = 1$ we need to consider the one sided limits:

$$\begin{aligned} C'(0) &= \lim_{t \to 0^+} C'(t) \\ &= \lim_{t \to 0^+} \sum_{i=0}^{n} \binom{n}{i} \cdot \Big(\underbrace{i \cdot t^{i-1} \cdot (1-t)^{n-i}}_{\to 0 \text{ for } i \neq 1} - \underbrace{t^i \cdot (n-i) \cdot (1-t)^{n-i-1}}_{\to 0 \text{ for } i \neq 0} \Big) \cdot P_i \\ &= -n \cdot P_0 + n \cdot P_1 = n \cdot (P_1 - P_0) \end{aligned}$$

$$\begin{aligned} C'(1) &= \lim_{t \to 1^-} C'(t) \\ &= \lim_{t \to 1^-} \sum_{i=0}^{n} \binom{n}{i} \cdot \Big(\underbrace{i \cdot t^{i-1} \cdot (1-t)^{n-i}}_{\to 0 \text{ for } i \neq n} - \underbrace{t^i \cdot (n-i) \cdot (1-t)^{n-i-1}}_{\to 0 \text{ for } i \neq n-1} \Big) \cdot P_i \\ &= -n \cdot P_{n-1} + n \cdot P_n = n \cdot (P_n - P_{n-1}) \end{aligned}$$

In both cases we again assumed that $0^0 := 1$. As we can see for $C'(0)$ only the summands with $i = 0$ and $i = 1$ contributed to the sum. For $C'(1)$ only the summands with $i = n-1$ and $i = n$ do. ■

3. Existence Of Interval Planar Drawings

In this chapter we will give a characterization of interval planar graphs and we will show that it is NP-complete to decide whether a given general graph and an interval assignment allow an interval planar embedding. The proof will be based on the known NP-completeness of upward planarity testing. In the remaining sections of this chapter we deal with special subclasses of graphs that allow upward planarity testing in polynomial time. We will give efficient interval planarity tests and embedding algorithms in some cases while showing NP-completeness in others.

3.1. Interval Consistency and Normal Form

We first take a closer look at the intervals assigned to the vertices. Let $\mathcal{I} \subseteq 2^{\mathbb{Z}}$ be the set of intervals over the integer numbers, let $G = (V, E)$ be a directed acyclic graph and let $I : V \to \mathcal{I}$ be a function that maps each vertex $v \in V$ to an interval of levels that v can be embedded to. We call this function I an *interval assignment* of the vertices V. A *level assignment* $l : V \to \mathbb{Z}$ maps each vertex to a level from its interval, so for every $v \in V$ we have $l(v) \in I(v)$. We further define $I_{min}(v) := \min I(v)$ and $I_{max}(v) := \max I(v)$ to be the lowest and highest possible level of vertex v.

Definition 3.1 An interval assignment I is *consistent*, if for each edge $(u, v) \in E$ levels $l_u \in I(u)$ and $l_v \in I(v)$ exist with $l_u < l_v$. Otherwise we call it *inconsistent*.

A graph with an inconsistent interval assignment does not have an interval planar embedding. There would be two vertices u and v connected by an edge $(u, v) \in E$ with $l_u \geq l_v$ for every $l_u \in I(u)$ and every $l_v \in I(v)$. Therefore the edge (u, v) cannot be embedded as a y-monotone curve, it cannot not end above its start. We see that a consistent interval assignment is a necessary condition for an interval planar embedding. However, in Figure 3.1a we see that a consistent interval assignment is not sufficient. In the given example it is possible for each edge independently to embed its start vertex below its end vertex. However combined this is not possible any more, because edge (x, y) forces vertex y to be an level 2 while edge (y, z) forces it to be on level 1. Our definition of consistency only checks the dependencies between two adjacent vertices. We are now going to extend this to also take all transitive dependencies into account.

(a) (b)

Figure 3.1.: (a) The shown interval assignment is consistent. However there is no interval planar embedding. (b) The same graph as on the left, but the interval assignment was transformed into normal form. Now it is obvious that there cannot be any interval planar embedding.

Algorithm 3.1: Construct Normal Form

Input: Graph $G = (V, E)$, interval assignment $I^{(1)}$.
Output: Interval assignment $I^{(2)}$ in normal form.

```
1 I^(2) ← I^(1)
2 forall v ∈ V in topological order do
3     forall (u, v) ∈ E do
4         I^(2)(v) ← I^(2)(v) ∩ [I^(2)_min(u) + 1, ∞)
5 forall v ∈ V in reverse topological order do
6     forall (v, w) ∈ E do
7         I^(2)(v) ← I^(2)(v) ∩ (−∞, I^(2)_max(w) − 1]
8 return I^(2)
```

Definition 3.2 An interval assignment I is in *normal form*, if the following two conditions hold for every edge $(u, v) \in E$:

$$I_{min}(u) < I_{min}(v) \quad \text{and}$$
$$I_{max}(u) < I_{max}(v)$$

Algorithm 3.1 describes a technique to transform an interval assignment $I^{(1)}$ of a graph $G = (V, E)$ into an interval assignment $I^{(2)}$ in normal form in $\mathcal{O}(|V|)$ steps. It works in two phases. The first one increases the lower bounds of the intervals. For each vertex $v \in V$ the lower bound must be strictly bigger than the lower bound of all its direct predecessors. So by traversing G in topological order all predecessors of the current vertex v already have their final lower bound assigned and we can determine v's final lower bound as well. The second phase decreases the upper bounds and works symmetrically to the first, traversing G in reverse topological order. Figure 3.1b shows the normalized intervals of the example in Figure 3.1a. They are all empty, so it immediately follows that there cannot be any interval planar embedding.

The intervals in the new interval assignment $I^{(2)}$ can be smaller than in $I^{(1)}$. However, we will now see that this does not make any otherwise possible embeddings impossible.

Lemma 3.3 *For a graph $G = (V, E)$ and an interval assignment $I^{(1)}$, the interval assignment $I^{(2)}$ in normal form as obtained from Algorithm 3.1 allows exactly the same interval planar embeddings as $I^{(1)}$.*

Proof Algorithm 3.1 produces an interval $I^{(2)}(v) \subseteq I^{(1)}(v)$ for every vertex $v \in V$. Therefore every interval planar embedding under $I^{(2)}$ is also a valid interval planar embedding under $I^{(1)}$.

To show that Algorithm 3.1 does not eliminate any interval planar embeddings while shrinking the intervals, we show that each shrinking operation of the algorithm does not reduce the set of all possible interval planar embeddings.

- Line 4: Assume this operation eliminates an embedding where vertices u and v are embedded at levels $l(u)$ and $l(v)$ respectively, connected by an edge (u, v). Since this embedding is eliminated, $l(v)$ must have been smaller than $I^{(2)}_{min}(u) + 1$, so $l(v) \leq I^{(2)}_{min}(u)$. Since u must be on a lower level than v, we get $l(u) < l(v) \leq I^{(2)}_{min}(u)$. This is a contradiction, so this would have not been a valid embedding in the first place.
- Line 7: Again we assume that the operation would eliminate a valid embedding with v on level $l(v)$ and w on level $l(w)$, connected by an edge (v, w). As above we get $l(v) \geq I^{(2)}_{max}(w)$. Since w must be on a higher level than v, we get $l(w) > l(v) \geq I^{(2)}_{max}(w)$. Again a contradiction, so this embedding was not in the first place. ∎

Lemma 3.3 allows us to only consider interval assignments in normal form in the following chapters. By doing so we will not loose any possible embeddings but we can safely ignore the consistency check for each edge, because all transitive dependencies are represented in the reduced intervals.

3.2. st-planar Graphs

Before looking at more general cases we will use this section to embed st-planar graphs, because they appear as subproblems in many other instances. Remember that those were graphs with only a single source s and a single sink t. As described by Di Battista [4], st-planar graphs are always upward planar. We will describe a method to choose a level for each vertex from its interval. Finding a planar embedding is then just a Level Planarity instance. By giving explicit constructions for polyline and straight line embeddings, we then show that this Level Planarity instance is always solvable with the obtained levels.

3.2.1. Level Assignment

The first step to find an interval planar embedding for an st-planar graph is to choose a level for each vertex. This choice is the same for polyline and straight line embeddings. How a topological embedding can always be obtained from these levels will be described in Sections 3.2.2 and 3.2.3.

Lemma 3.4 *Let $G = (V, E)$ be an st-planar graph and I be an interval assignment in normal form. If $I(v) \neq \emptyset$ for all $v \in V$, we can set $l(v) = I_{min}(v)$ and we can obtain an interval planar embedding with these levels.*

Proof With $l(v) = I_{min}(v)$ we obviously get $I_{min}(v) \leq l(v) \leq I_{max}(v)$ for all $v \in V$, so the assignment is not in conflict with the intervals. We still need to show that this assignment gives an upward embedding, meaning that for each edge $(u, v) \in E$ we get $l(u) < l(v)$. This follows from the fact, that the interval assignment I is in normal form, so we have $l(u) = I_{min}(u) < I_{min}(v) = l(v)$ for each edge $(u, v) \in E$.

We have not yet proved that this level assignment actually has a level planar embedding. To avoid repetition, we refer to the proofs of Theorem 3.5 and Theorem 3.8 that will close this gap in two different ways. ∎

In Lemma 3.4 we chose the levels greedily as low as possible. Any interval planar embedding based on this level assignment will therefore have the y-coordinates of all vertices at the lowest possible position.

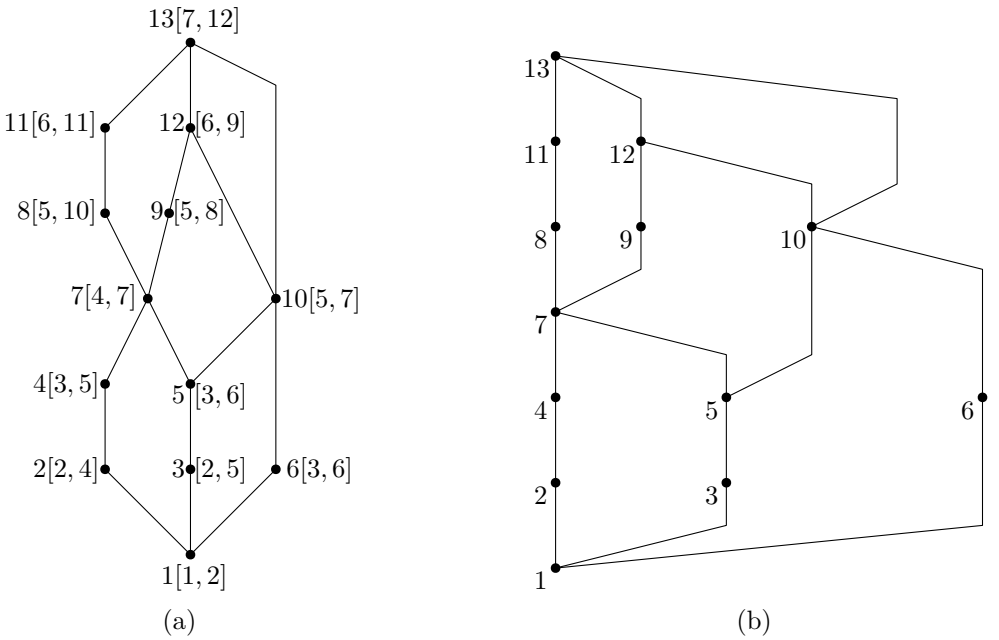

Figure 3.2.: (a) An upward planar embedding of some sample graph. (b) The graph from the left with a polyline interval planar embedding as constructed via Theorem 3.5.

3.2.2. Polyline Drawing

After assigning levels to the vertices we can construct a polyline planar embedding with respect to these levels.

Theorem 3.5 *Let $G = (V, E)$ be an st-planar graph with a level assignment l obtained from an interval assignment in normal form via Lemma 3.4. Then G has a polyline interval planar embedding where each vertex has its level as its y-coordinate. The number of edge bends is in $\mathcal{O}(|V|)$.*

Proof Take any upward planar embedding Γ of G (recall that Γ exists for all st-planar graphs). Based on a left-to-right ear-decomposition $E_1, \ldots, E_k$ we are now going to construct a polyline embedding of G in several steps. Starting with an empty drawing, we extend the current drawing in each step by the next ear in the ear-decomposition. To illustrate the construction we will embed the upward planar graph in Figure 3.2a parallel to this proof obtaining the embedding shown in Figure 3.2b.

The first ear E_1 is always an st-path. Draw it on a vertical line with each vertex $v \in V(E_1)$ at coordinates $(1, l(v))$. In our running example this is the path $1 \to 2 \to 4 \to 7 \to 8 \to 11 \to 13$. For each following ear E_i call the first vertex E_i^s and the last vertex E_i^t. They are already positioned by earlier ears. We need to distinguish two cases:

- If E_i has no inner vertices, we draw a polyline from E_i^s to E_i^t with two bends at $(i, l(E_i^s) + 0.5)$ and $(i, l(E_i^t) - 0.5)$. In our example this is the case for ear $10 \to 13$.
- If E_i has inner vertices v, we draw them at $(i, l(v))$ and connect them with vertical edges. We connect E_i^s with the first inner vertex via a polyline with one bend at position $(i, l(E_i^s) + 0.5)$. Similarly we connect the last inner vertex with E_i^t via a polyline with one bend at position $(i, l(E_i^t) - 0.5)$. In Figure 3.2b this is the case for the remaining ears: $1 \to 3 \to 5 \to 7$ and $5 \to 10 \to 12$ are just two of them.

For each but the first ear we introduced two edge bends. Since the number of ears is bounded by the number of edges, the total number of bends is less than $2|E|$. Since G is planar, this is in $\mathcal{O}(|V|)$.

Algorithm 3.2: Polyline Embedding for st-Planar Graphs

Input: Graph $G = (V, E)$, st-planar.
Output: Polyline interval planar embedding Γ of G.

$E_1, \ldots, E_k \leftarrow$ left-to-right ear-decomposition of G
`// Embed first, leftmost ear.`
forall $v \in V(E_1)$ **do**
  $\Gamma \leftarrow \Gamma \cup \{v$ at coordinates $(1, l(v))\}$.
forall $(u, v) \in E(E_1)$ **do**
  $\Gamma \leftarrow \Gamma \cup \{$line: $u \to v\}$
`// Embed remaining ears.`
for $i = 2$ **to** k **do**
  $n \leftarrow V(E_i)$
  $v_1, \ldots, v_n \leftarrow$ vertices of E_i from bottom to top
  `// First and last edges, the ones with bends.`
  if $n > 2$ **then**
    $\Gamma \leftarrow \Gamma \cup \{$line: $v_1 \to (i, l(v_1) + 0.5) \to v_2\}$
    $\Gamma \leftarrow \Gamma \cup \{$line: $v_{n-1} \to (i, l(v_n) - 0.5) \to v_n\}$
  else
    $\Gamma \leftarrow \Gamma \cup \{$line: $v_1 \to (i, l(v_1) + 0.5) \to (i, l(v_2) - 0.5)) \to v_2\}$
  `// Inner vertices and edges.`
  for $j = 2$ **to** $n - 1$ **do**
    $\Gamma \leftarrow \Gamma \cup \{v_j$ at coordinates $(i, l(v_j))\}$.
  for $j = 2$ **to** $n - 2$ **do**
    $\Gamma \leftarrow \Gamma \cup \{$line: $v_j \to v_{j+1}\}$
return Γ

We still need to show that the resulting embedding is planar. We do this by induction on the number of ears already added. Remember that we called the rightmost st-path in $G_i = \bigcup_{j=1}^{i} E_j$ the right front of G_i.

The induction base is the first ear E_1. It gets drawn as a vertical line, obviously planar and no intersections are possible.

Every other ear E_i $(2 \leq i \leq k)$ can be seen as the union of three line segments. The middle one is a vertical line with a greater x-coordinate than all older segments from ears $E_1, \ldots, E_{i-1}$. No intersection is possible here. The first segment starts at the right front of G_{i-1}, so a horizontal ray to the right would not intersect any other segment. Anyway, the first segment is not a horizontal line, but goes up by 0.5 units on its way. All other outgoing segments from E_i^s make the same vertical difference but on a smaller horizontal difference. Therefore they have a bigger slope and we have no intersection except at E_i^s. The third segment can be treated analogously to the first one (with negative vertical difference -0.5). ∎

The construction algorithm given in the proof of Theorem 3.5 is given in pseudocode in Algorithm 3.2.

Corollary 3.6 *Algorithm 3.2 constructs a polyline interval planar embedding for any st-planar graph G requiring area in*

$$\mathcal{O}\left(|V| \cdot (l(t) - l(s))\right).$$

PROOF The height is $\max_{v \in V} l(v) - \min_{v \in V} l(v)$, because each vertex gets its assigned level as a y-coordinate. But because the drawing is upward and an st-planar graph G has a single source s and a single sink t, we know that $\max_{v \in V} l(v) = l(t)$ and that $\min_{v \in V} l(v) = l(s)$. Further, consecutive ears are positioned at consecutive integer x-coordinates. We know that the number of ears is bounded by $\mathcal{O}(|V|)$. ∎

Corollary 3.7 *Algorithm 3.2 constructs a polyline interval planar embedding of any st-planar graph $G = (V, E)$ in $\mathcal{O}(|V|)$ time.*

PROOF According to Lemma 2.9 the ear-decomposition $E_1, \ldots, E_k$ can be obtained in $\mathcal{O}(|V|)$. Ear E_1 is a path that is embedded on a vertical line. This is possible in $\mathcal{O}(|V(E_1)|)$. For each following ear E_i $(2 \leq i \leq k)$, the construction needs to assign positions to $|V(E_i)| - 2$ vertices (the first and the last vertex have already been embedded) and $|E(E_i)| - 2$ line segments. This can be done in $\mathcal{O}(|V(E_i)|)$ time. In total this gives us $\mathcal{O}(|V|)$ time. ∎

3.2.3. Straight Line Drawing

In the last section we constructed relatively compact polyline drawings. However it is also possible to find x-coordinates to the assigned levels such that we get a straight line drawing.

Theorem 3.8 *Let $G = (V, E)$ be an st-planar graph with a level assignment l obtained from an interval assignment in normal form via Lemma 3.4. Then G has a straight line interval planar embedding where each vertex has its level as its y-coordinate.*

PROOF Again we will construct an interval planar embedding via a left-to-right ear-decomposition $E_1, \ldots, E_k$ of G and the result of this construction is shown in Figure 3.4. In each step we add the next ear E_i to the already drawn subgraph $G_{i-1} \subseteq G$. To get a straight line embedding, we make sure the following invariant is fulfilled during every step i $(1 \leq i \leq k)$ of the algorithm: All non-adjacent vertices at the right front of G_i are visible to each other. Two vertices u and v are *visible* to each other, if drawing a straight line between them does not intersect any other vertex or edge in G_i.

The first ear E_1 is an st-path. Let $e_1, \ldots, e_{|E(E_1)|}$ be the edges on E_1 from s to t. We are going to choose the x-coordinates of the vertices to fulfill the visibility invariant: For every edge $e_i = (u, v) \in E_1$ $(1 < i < |E(E_1)|)$, we choose the x-coordinate of u strictly smaller than the x-coordinate of v. This gives the straight line between u and v a positive slope $s_{(u,v)}$. Further we require that the slopes along E_1 are strictly decreasing. This way the vertices of E_1 form the vertices of a convex polygon, therefore the visibility invariant is fulfilled. In Figure 3.4 we can see such an embedding of $E_1 = (1 \to 2 \to 4 \to 7 \to 8 \to 11 \to 13)$.

For every other ear E_i $(2 \leq i \leq k)$ we now need to find x-coordinates that keep the visibility invariant fulfilled. One possible way to do so, is to consider the first and the last point on E_i, which we will again denote with E_i^s and E_i^t respectively. Take these two and all vertices between them on the right front of G_{i-1} and call this set of vertices C_i. Now let C_i be the control points of a Bézier curve B_i. For some vertex $v \in V(E_i)$ we can now choose the x-coordinate at which B_i intersects the level $l(v)$. This process is shown in Figure 3.3 for some sample ear E_i.

To justify that the chosen x-coordinates are valid we need to use some properties of Bézier curves. Let B_i be the Bézier curve with control points $C_i = \{C_1, \ldots, C_k\}$.

Planarity. The resulting embedding is planar, because B_i intersects exactly its first and last control points C_1 and C_k an no others by Lemma 2.12. By Lemma 2.11 we even know

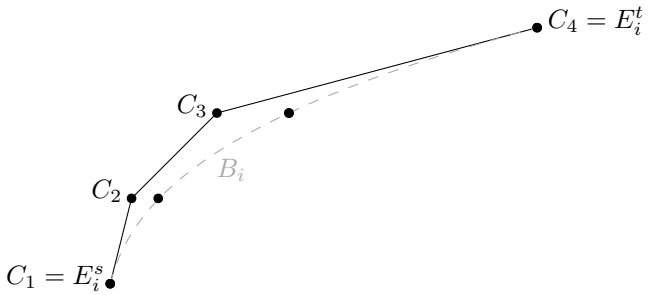

Figure 3.3.: How to find valid x-coordinates for an ear E_i to keep the visibility invariant fulfilled: The set of control points is given by $C_i = \{C_1, C_2, C_3, C_4\}$. The vertices on ear E_i are embedded at the unique x coordinate that curve B_i intersects their level at.

that B_i lies completely within the convex hull of C_i, so each E_i is embedded below the polyline induced by the right front of G_{i-1}, just intersecting it at its endpoints.

Uniqueness. Above we used that curve B_i uniquely determines the x-coordinate for each vertex on E_i. This is true, because B_i is strictly monotone increasing in the vertical direction if its control points C_i are. This guarantees a unique intersection of B_i and the horizontal line at $l(v)$ for each $v \in V(E_i)$.

Visibility Invariant. Since the control points C_i formed a convex polygon, curve B_i is convex as well, meaning that a straight line segment between any two points on B_i would not intersect B_i. So the vertices of E_i that are embedded on B_i fulfill the visibility invariant. We still need to show, that they do so together with all vertices of the right front of G_i as well. The tangent of B_i at some point p converges to the line defined by C_1 and C_2, the closer p gets to C_1 by Lemma 2.13. Similarly it converges to the line defined by C_{k-1} and C_k, if p approaches C_k (also by Lemma 2.13). This guarantees that the visibility invariant is not broken at $C_1 = E_i^s$ and at $C_k = E_i^s$ and is therefore valid everywhere on the right front. ■

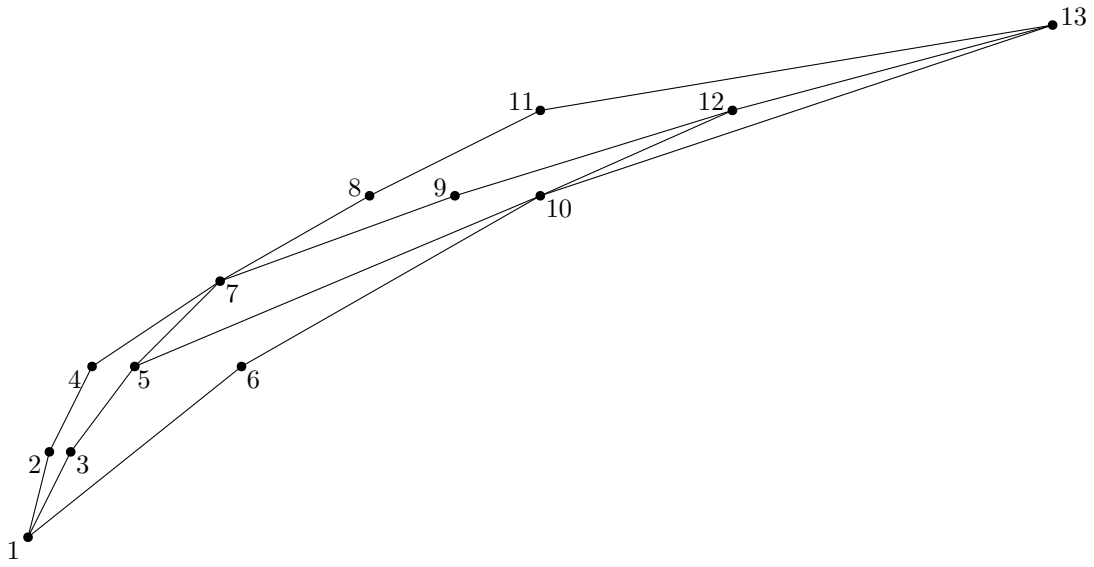

Figure 3.4.: A straight line embedding as constructed via Theorem 3.8. The graph used is the same as for the polyline embedding in Figure 3.2.

The construction algorithm given in the proof of Theorem 3.8 is given in pseudocode in Algorithm 3.3. Worth mentioning is how the x-coordinates for the first ear are computed in Lines 6 and 7, because we did not provide a concrete formula in above proof. We said that the coordinates need to be chosen in order to give decreasing positive slopes on the edges of E_1. In the pseudocode we set the horizontal distance of two consecutive vertices on E_1 to a multiple of their vertical difference, so that the first slope is 1, the second is 1/2, the third is 1/3 and so on.

The straight line drawings constructed with Algorithm 3.3 are not really readable. They will always have a similar shape and the edges can be embedded very close to each other. They can also require an exponential amount of area, since there is upward planar graphs that need an exponential amount of area for all their straight line drawings as shown by Di Battista [5].

When constructing a straight line drawing, the most time in Algorithm 3.3 is spent finding the x-coordinates of the vertices, because evaluating or approximating the Bézier curve takes linear time in the number of control points. While the construction using the Bézier curves allowed a very elegant proof for the existence of such drawings, we can speed up the actual construction to run in $\mathcal{O}(|V|)$ time.

Corollary 3.9 *It is possible to construct a straight line interval planar embedding of any st-planar graph $G = (V, E)$ in $\mathcal{O}(|V|)$ time.*

Proof All operations in Algorithm 3.3 need only constant time per vertex or edge. The only exception is the computation of the x-coordinates of the vertices. When embedding an

Algorithm 3.3: Straight Line Embedding for st-Planar Graphs

Input: Graph $G = (V, E)$, st-planar.
Output: Straight line interval planar embedding Γ of G.

1 $E_1, \ldots, E_k \leftarrow$ left-to-right ear-decomposition of G

`// Embed first, leftmost ear.`
2 $v_1, \ldots, v_{|E_1|} \leftarrow$ vertices of E_1 from bottom to top.
3 $X \leftarrow 0$
4 $\Gamma \leftarrow \Gamma \cup \{\text{vertex } v_1 \text{ at coordinates } (X, l(v_1))\}$
5 **for** $i = 2$ **to** $|E_1|$ **do**
6 $\quad \Delta \leftarrow l(v_i) - l(v_{i-1})$
7 $\quad X \leftarrow X + (i-1) \cdot \Delta$
8 $\quad \Gamma \leftarrow \Gamma \cup \{\text{vertex } v_i \text{ at coordinates } (X, l(v_i))\}$
9 $\quad \Gamma \leftarrow \Gamma \cup \{\text{line: } v_{i-1} \to v_i\}$

`// Embed remaining ears.`
10 **for** $i = 2$ **to** k **do**
11 $\quad v_1, \ldots, v_{|E_i|} \leftarrow$ vertices of E_i from bottom to top.
12 $\quad C_1, \ldots, C_n \leftarrow$ vertices on right front of Γ between v_1 and $v_{|E_1|}$
13 $\quad B \leftarrow$ Bézier curve with control points $C_1, \ldots, C_n$
14 $\quad$ **for** $j = 2$ **to** $|E_i| - 1$ **do**
15 $\quad\quad X \leftarrow x$-coordinate of B's intersection with horizontal $y = l(v_i)$
16 $\quad\quad \Gamma \leftarrow \Gamma \cup \{\text{vertex } v_i \text{ at coordinates } (X, l(v_i))\}$
17 $\quad$ **for** $j = 2$ **to** $|E_i|$ **do**
18 $\quad\quad \Gamma \leftarrow \Gamma \cup \{\text{line: } v_i \to v_{i+1}\}$
19 **return** Γ

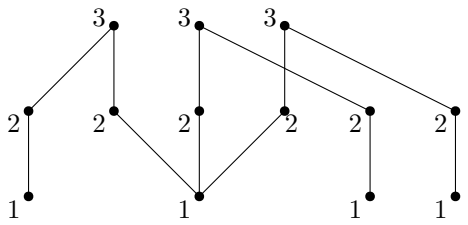

Figure 3.5.: An upward planar tree that is not level planar with the assigned levels. Since level planarity is just a special case of interval planarity this tree is also not interval planar with the given levels (one element intervals).

ear E_i $(2 \leq i \leq k)$, we can embed all inner vertices of $E_i = \{v_1, \ldots, v_n\}$ on the straight line between its endpoints v_1 and v_n (which are already embedded). This gives those vertices initial x-coordinates, but if the ear has any inner vertices, these x-coordinates violate the visibility invariant. This can be fixed by moving the vertices to the left. Start with v_2, it can be shifted left just shortly before it would intersect the right front of G_{i-1}. Then edge (v_1, v_2) still has a lower slope than the preceding one on the right front of G_i and a greater slope that the rest of ear E_i. Next we shift v_3 to the left until just before the slope of edge (v_2, v_3) would become equal to the slope of the preceding edge (v_1, v_2). Doing this for all vertices on ear E_i restores the visibility invariant. ■

3.3. Characterization of Interval Planar Graphs

We will use this section to present a combinatorial characterization of interval planar graphs. This extends a characterization of upward planar graphs to also take the interval assignment into account.

Di Battista [4] showed that a directed graph G is upward planar, if and only if it is the spanning subgraph of an st-planar graph. Of course, upward planarity is a necessary condition. However it is not sufficient as Figure 3.5 shows. In Section 3.1 we saw that a consistent interval assignment is another necessary but not sufficient condition.

Theorem 3.10 *Let $G = (V, E)$ be a directed graph and let I be an interval assignment. The following statements are equivalent:*

1. *G is interval planar.*
2. *G is a subgraph of an st-planar graph $G' = (V \dot\cup \{s, t\}, E')$ with an interval assignment I' in normal and $\emptyset \neq I'(v) \subseteq I(v)$ for all $v \in V$ and $I'(s), I'(t) \subseteq \mathbb{Z}$.*

Proof We start by showing that Statement 2 implies Statement 1. If $G' = (V \dot\cup \{s, t\}, E')$ is an st-planar graph and I' an interval assignment in normal form with non-empty intervals, we can find an interval planar embedding as described in Section 3.2. By deleting the vertices s, t and all edges $e \in E' \setminus E$ we get an interval planar embedding of G. Since $I'(v) \subseteq I(v)$ for all $v \in V$, the embedding is valid for G under I and G is therefore interval planar.

To show that Statement 1 implies 2, let $G = (V, E)$ be interval planar under a consistent interval assignment I. Consider an arbitrary interval planar embedding Γ of G. To augment G to an st-planar graph G' we need to extend it by supersource s and supersink t and add dummy edges until we s and t are the only source and the only sink. Define

$$S^{min} := \{v \in V \mid v \text{ is a source with minimal } y\text{-coordinate}\} \quad \text{and}$$
$$T^{max} := \{v \in V \mid v \text{ is a sink with maximal } y\text{-coordinate}\}$$

to be the lowest sources and highest sinks in Γ. Denote the corresponding y-coordinates with y_{min} and y_{max} respectively. We first extend the vertex set by a supersource s and a supersink t, so $V' := V \dot\cup \{s, t\}$ and define I' as follows:

$$I'(v) := \begin{cases} [y_{min} - 1, y_{min} - 1] & \text{if } v = s \\ [y_{max} + 1, y_{max} + 1] & \text{if } v = t \\ [l(v), l(v)] & \text{if } v \in V, \text{ where } l(v) \text{ is the level of } v \text{ in } \Gamma \end{cases}$$

We can then extend the embedding Γ by placing s at y-coordinate $y_{min} - 1$ and t at y-coordinate $y_{max} + 1$ and connect s with all sources in S^{min} and all sinks in T^{max} with t. Embedding Γ can have several more sources and sinks that we can augment with a method described in [4]: Start with $E' = E$. For a sink $t' \neq t$, draw a straight edge e' leaving t' upward until shortly before it would intersect another edge. Then follow this edge closely to its end. If the upward edge would not intersect any other edge, then t' can directly be connected to the supersink t. Add the new edge e' to E'. A source $s' \neq s$ can be augmented similarly by drawing an edge downward.

Adding dummy edges to reduce the number of sources and sinks was done without breaking planarity. What is left to show is that I' is an interval assignment in normal form, even after adding edges in the previous step. All intervals in I' contain only a single integer, for simplicity we will also denote it by $I(v)$ for each $v \in V$ in the following. To show that I' is in normal form we need $I(u) < I(v)$ for every edge $(u, v) \in E'$. This is obviously true for all $e \in E$, because we started with an interval planar embedding. For all $e \in E' \setminus E$, this is true as well, because they are upward by the described construction and therefore their endpoint has a y-coordinate strictly greater than their startpoint. These y-coordinates correspond with the one-element intervals of the vertices. ∎

We are going to use Theorem 3.10 several times in later sections. It tells us, that if we want to find an interval planar embedding of a given graph, we need add a supersource s, a supersink t and dummy edges until no other sources and sinks are left. The interval assignment however must not contain empty intervals for any vertex when converted to normal form. As it turns out, this can efficiently be done for some graph classes while it is NP-complete for others. When we transformed the graph into an st-planar graph, we can use the known methods from Section 3.2 to embed st-planar graphs.

We summarize this as a general embedding strategy for a graph $G = (V, E)$, consisting of three steps:

1. Augment G by adding a supersource s, a supersink t and dummy edges to a graph G', so that G' is st-planar and has an interval assignment in normal form with non-empty intervals. If this is impossible, no interval planar embedding exists and we can stop here.
2. Use the methods from Section 3.2 to get an interval planar embedding Γ of G'.
3. Remove supersource s, supersink t and all dummy edges from G' to obtain an interval planar embedding of G.

Step 1 is the hard part of this method. Consider a sink vertex $t' \in V$ in G. To augment G to an st-planar graph, we need to add a dummy edge from t' to another vertex $v \in (V \cup \{t\})$. Every such v is a possible candidate. However, if we already have a given combinatorial embedding we can observe that all outgoing edges of t' would lie inside some face f. This reduces the candidates to the vertices on the boundary of f. Actually we can go even further: We only need to consider the *face sinks* of f, that is all vertices on f's boundary that do not have any outgoing edges incident to f. This is justified by the fact that all

other vertices on f's boundary will definitely be embedded below some face sink of f. We can make a similar observation for sources s'. If a combinatorial embedding is given, we only need to consider the *face sources* as candidates to cancel s'.

We will refer to the process of adding dummy edges between sources or between sinks as *source canceling* and *sink canceling* from now on. When we know how to efficiently do this, we already know how to find an interval planar embedding.

3.4. General Graphs

Garg and Tamassia [8] showed that testing for upward planarity is NP-complete. We can use their result to easily show the NP-completeness of testing for interval planarity.

Theorem 3.11 Interval Planarity *is NP-complete, i.e. for a directed acyclic graph $G = (V, E)$ and an interval assignment I, it is NP-complete to decide, whether there is an interval planar embedding of G.*

Proof Interval Planarity is in NP, since we can easily check a given embedding. We just need to test that it is planar and that we have $I_{min}(v) \leq l(v) \leq I_{max}(v)$ for each vertex $v \in V$.

Now consider a directed acyclic graph $G = (V, E)$. We define $I(v) := [1, |V|]$ for each $v \in V$. If there is an interval planar embedding of G under I, this is also an upward planar embedding. If there is no interval planar embedding of G under I, then G is not upward planar. This is because no upward planar embedding can have more than $|V|$ different y-coordinates for the vertices which could then be injectively mapped to the allowed levels in $[1, |V|]$. ∎

By Theorem 3.11 we see that unless $P = NP$, we cannot give an efficient algorithm to test a given graph for interval planarity. We will spend the remainder of this chapter looking at several special graph classes for which efficient upward planarity tests exist and explore, for which of them we can also check interval planarity efficiently.

3.4.1. Fixed Combinatorial Embedding

Bertolazzi et al. [2] showed that general directed graphs can be tested for upward planarity under a fixed combinatorial embedding in polynomial time. However, for interval planarity the fixation of an embedding does not allow an efficient test. We will show NP-completeness via a reduction starting from Planar Monotone 3SAT, which is NP-complete as shown by De Berg and Khosravi in 2012 [3].

Definition 3.12 Planar Monotone 3SAT

Instance Given a 3SAT formula φ together with a planar rectilinear embedding Γ. Every clause in φ contains exaclyt three literals, either all positive or all negative.

Question Is there a truth assignment of the literals such that φ is `true`.

Figure 3.6 shows an example of a Planar Monotone 3SAT instance. It represents a boolean formula in conjunctive normal form where each clause contains exactly three literals. The variables are shown as segments on a horizontal line and the clauses are shown as E-shapes connecting three variables each, either above or below them. A nice property of Planar Monotone 3SAT is that we can embed the clause shapes in a way, that all positive clauses (the ones containing only positive literals) are below the variables and all negative clauses are above it. This way we do not need to further distinguish which E-shape represents a positive and which a negative clause.

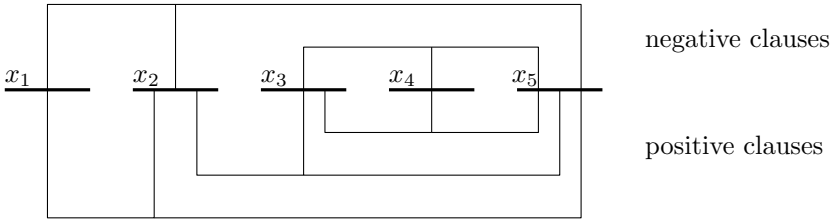

Figure 3.6.: An instance of Planar Monotone 3SAT. In this rectilinear embedding all positive clauses are below the variables and all negative clauses are above the variables. This instance corresponds to the following boolean formula: $(x_1 \vee x_2 \vee x_5) \wedge (x_2 \vee x_3 \vee x_5) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_5}) \wedge (\overline{x_3} \vee \overline{x_4} \vee \overline{x_5})$

Theorem 3.13 Interval Planarity *is NP-complete even if the graph G has a fixed combinatorial embedding and outer face.*

Proof We already know from Theorem 3.11 that Interval Planarity is in NP. It is so as well when we fix the combinatorial embedding and the outer face.

Given a Planar Monotone 3SAT instance φ with a planar rectilinear embedding Γ. In Γ, every positive clause is embedded below the variables and every negative clause is embedded above the variables as shown in Figure 3.6. We are going to transform this into an instance of Interval Planarity. Replace each E-shape clause in Γ with a gadget as shown in Figure 3.7: Each gadget has three vertices $x_{i,1}$, $x_{i,2}$ and $x_{i,3}$ that correspond to the three literals that appear in the clause. The gadgets of two clauses that both contain the same variable are merged at the corresponding vertices. Since Γ was planar and each gadget mimics the E-shape, the resulting graph also has a planar embedding.

For an interval planar embedding we need to cancel the sources and sinks in each of the gadgets. Consider the gadget corresponding to the i-th clause. Without loss of generality this is a positive gadget (corresponding to a positive clause). It contains a sink p_i, that we call the *pendulum* of the gadget. There are three different sinks in the gadget, that p_i can be canceled to: $x_{i,1}$, $x_{i,2}$ and $x_{i,3}$, each corresponding to one of the variables in φ. In Figure 3.7 the three choices are shown by the gray dashed lines. We are going to show, that finding a valid assignment of all p_i's to one of their three possible sinks (or sources in negative gadgets) is equivalent to finding a satisfying variable assignment for the Planar Monotone 3SAT instance φ.

First observe, that the level of each pendulum p_i is fixed to 0 in all gadgets. Therefore, in a positive clause i the sink that p_i gets connected to must be at level 1. Symmetrically, in a negative clause j the source that p_j gets connected to will be at level -1. So if a positive gadget i and a negative gadget j share a literal, their pendulums p_i and p_j cannot both be canceled with the same vertex (a sink in gadget i and a source in gadget j).

Assume an interval planar embedding of G is given. This induces an assignment for the pendulums p_i for each gadget. We can now extract a truth assignment for φ from this: For the interval planar embedding each pendulum p_i was canceled with some vertex $x_{i,k}$ that corresponds to the k-th variable in φ. If p_i was in a positive gadget, set variable x_k to `true`. If p_i was in a negative gadget, set x_k to `false`. By above observation it is impossible that a vertex corresponding to a variable x_k is simultaneously set to `true` and to `false`. If any variable is not yet set, we can either set it to `true` or `false`. The extracted variable assignment is indeed a truth assignment, because each gadget contains a pendulum and each pendulum sets a literal in the corresponding clause to `true`.

If on the other hand a truth assignment of φ is given, we can use it to assign the pendulums p_i in each gadget i to one of the three possible sources or sinks. Each clause contains at

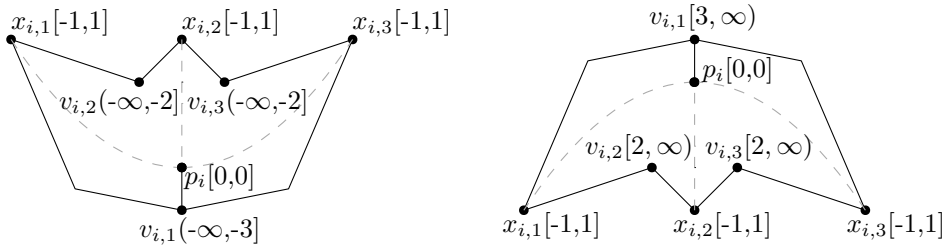

Figure 3.7.: The gadgets to replace the Planar Monotone 3SAT clauses with. The left gadget replaces each positive clause below the variables, the right gadget replaces each negative clause above the variables. The dashed lines show the possible ways to cancel the pendulum p_i.

least one literal that was assigned the value `true`. Use a vertex of gadget i corresponding to one of these true literals for the pendulum p_i. Since no variable was both `true` and `false` at the same time, no vertex that corresponds to a variable will get a pendulum from a positive and a negative gadget assigned at the same time. Therefore an interval planar embedding exists for this way of canceling the pendulums. ∎

This reduction works on a fixed embedding of the graph that we construct from the Planar Monotone 3SAT instance. The only variations we can force on the embedding are the dummy edges to cancel the inner sources and inner sinks. This implies that Interval Planarity is already NP-complete for graphs with inner sources and sinks under a fixed combinatorial embedding.

The only way to further restrict this is to restrict the number of sources or sinks to one. If both are restricted, we get an st-planar graph. In section 3.2 we saw an efficient test and embedding algorithm for them. If only one type is forbidden, we get a single source (or single sink) graph handled next in section 3.5.

Remark 3.14 The vertices corresponding to the variables all have interval $[-1, 1]$ in all gadgets of Figure 3.7. We can prove Theorem 3.13 with similar gadget where each variable vertex has interval $[0, 1]$. This would have the advantage, that we get a direct correspondence between the assigned level in an interval planar embedding and a truth value (0 is `false` and 1 is `true`). This is more elegant that in the given proof above where we say that 1 is `true`, -1 is `false` and that 0 can be arbitrarily chosen. However, a disadvantage is that the gadgets for positive and negative clauses are not symmetric any more: The gadget for positive clauses would remain unchanged (except for the smaller intervals at $x_{i,1}$, $x_{i,2}$ and $x_{i,3}$). In the gadget for negative clauses, the pendulum p_i now gets interval $[1, 1]$, so the source that it gets canceled with is forced to be embedded at level 0 instead of level -1.

Remark 3.15 Using a variation of the gadgets shown in Figure 3.7 we can even show the NP-completeness for graphs with a fixed embedding and an interval assignment with $|I(v)| \leq 2$ for each vertex $v \in V$: The pendulums have $|I(p_i)| = 1$ anyway. For the vertices corresponding to a variable we take the same approach as in Remark 3.14 and get $|I(x_{i,j})| \leq 2$. For all other vertices we can assign one-element intervals. To do this, we start with the rectilinear embedding of the Planar Monotone 3SAT instance as shown in Figure 3.6. This tells us how the gadgets are nested. We start assigning y-coordinates for the innermost gadgets. For example a positive innermost gadget i will get $[-3, -3]$ for vertex $v_{i,1}$ and $[-2, -2]$ for the two remaining vertices $v_{i,2}$ and $v_{i,3}$. A second innermost positive gadget j will then be assigned y-coordinates just below that. The two vertices $v_{j,2}$ and $v_{j,3}$ will get $[-4, -4]$, the remaining vertex $v_{j,1}$ will get $[-5, -5]$. We continue like this until each vertex has an interval assigned.

3.5. Single Source Graphs

Hutton and Lubiw [10] first gave an $\mathcal{O}(|V|^2)$ upward planarity test for single source directed acyclic graphs. Later Bertolazzi, Di Battista, Mannino and Tamassia [1] improved this to an optimal $\mathcal{O}(|V|)$ algorithm. For a given graph, both describe a method to test for upward planarity with the same combinatorial embedding and outer face. Then they decompose the graph into its triconnected components to find an upward planar embedding with an arbitrary combinatorial embedding. Similarly we will give an $\mathcal{O}(|V|)$ algorithm to check a single source directed acyclic graph for interval planarity with the same combinatorial embedding and outer face. However, we will see that it is NP-complete to test for interval planarity without fixing a combinatorial embedding.

Note that any result applying to single source graphs also applies to single sink graphs. Those have only one sink but might have several sources. Single sink instances can be mapped bijectively to single sources instances as shown in the following lemma:

Lemma 3.16 *Let G_t be a directed acyclic graph with a single sink and an interval assignment I_t. Then we can augment G_t and I_t to a graph G_s with a single source that has an interval planar embedding under an interval assignment I_s, if and only if G_t has an interval planar embedding under I_t.*

Further, if it exists, an embedding for G_t under I_t can be obtained from an embedding of G_s under I_s.

Proof Let $G_t = (V, E_t)$ be a directed acyclic graph with a single sink t and let I_t be an interval assignment. Define $G_s = (V, E_s)$ to be a graph on the same vertex set as G_t, where $E_s = \{(u, v) \mid (v, u) \in E_t\}$ contains all edges of E_t in their reverse direction. Now let C be a constant such that $C > \max \bigcup_{v \in V} I_t(v)$, so C is bigger than any element in any interval of I_t. For each vertex $v \in V$ we define $I_s(v) := C - I_t(v) := \{C - x \mid x \in I_t(v)\}$.

Vertex t is the single source of G_s, so G_s is indeed a single source graph. For each edge $(u, v) \in E_t$ there is exactly one edge $(v, u) \in E_s$. For every vertex v, intervals $I_t(v)$ and $I_s(v)$ have the same size. If u and v are two adjacent vertices, their intervals have the same relative position in I_s and I_t. The difference between their minimal and their maximal elements is the same:

$$\begin{aligned} \min I_s(u) - \min I_s(v) &= (C - \min I_t(u)) - (C - \min I_t(v)) \\ &= \min I_t(v) - \min I_t(u) \end{aligned}$$

$$\begin{aligned} \max I_s(u) - \max I_s(v) &= (C - \max I_t(u)) - (C - \max I_t(v)) \\ &= \max I_t(v) - \max I_t(u) \end{aligned}$$

From this we can now follow that an interval planar embedding of G_t under I_t exists, if and only if an interval planar embedding of G_s under I_s exists. Let Γ be such an interval planar embedding of G_s under I_s. For each edge $(u, v) \in E_s$ we know that if v is placed k levels higher than u, then the intervals in I_t allow us to place u also k levels above v in an embedding of G_t. So it is possible to assign valid levels to all vertices: Set $l(t) := \max I_t(t)$ to the single sink t. Each other vertex v gets $l(v)$ assigned, such that the level difference to t is equal to the level difference of v and t in G_s. With the same procedure an interval planar embedding of G_t under I_t can be converted into an interval planar embedding of G_s under I_s.

Since the underlying undirected graph is the same for G_s and G_t, we also get planarity for an embedding of G_t. ∎

3.5.1. Fixed Combinatorial Embedding

Throughout this section $G = (V, E)$ is a directed acyclic graph with a single source vertex $s \in V$ and an interval assignment I in normal form. We further assume that G is upward planar, since this is a necessary condition for interval planarity. Let Γ be any combinatorial embedding of G. We know that all these conditions can be checked in $\mathcal{O}(|V|)$ and will describe a method to test for interval planarity under I with the same combinatorial embedding and outer face as in Γ.

An inner sink is a sink not incident to the outer face, so it needs to be canceled with some sink of the face it lies in. The key observation is that every face in an embedded single source graph has only a single sink. So for any inner sink of the graph there is only one possible candidate to cancel it with. If we do so and add the corresponding dummy edges, the resulting graph has all its remaining sinks on the boundary of the outer face and thus they can all be canceled with a supersink t. This gives us an st-planar graph that can then be embedded with the known methods from Section 3.2. This process is given in pseudocode in Algorithm 3.4.

Theorem 3.17 *Let $G = (V, E)$ be a single source directed graph and let I be an interval assignment in normal form. Then we can construct an interval planar embedding of G in $\mathcal{O}(|V|)$ or detect that none exists.*

Proof The number of sinks in G is bounded by $\mathcal{O}(|V|)$ and each of them can be canceled with the unique sink of the face containing it in constant time. Further, adding the supersink and dummy edges from each remaining sink also takes constant time per sink. The sink canceling thus takes $\mathcal{O}(|V|)$ in total.

Running Algorithm 3.1 to get a new interval assignment for the resulting st-planar graph also takes time in $\mathcal{O}(|V|)$ and we know that we can find an interval planar embedding for an st-planar graph in $\mathcal{O}(|V|)$ from Section 3.2.2. All combined, Algorithm 3.4 runs in $\mathcal{O}(|V|)$. ∎

Algorithm 3.4: Embed Single Source Graph

Input: Directed, acyclic graph $G = (V, E)$, interval assignment I in normal form, combinatorial embedding Γ and fixed outer face.
Output: Γ' interval planar embedding of G or `NO EMBEDDING` if none exists.

```
// Cancel all inner sinks.
forall inner sinks t ∈ V do
    t_f ← sink of face f that any edge leaving t to the top lies in
    E ← E ∪ (t, t_f)
// Add supersink t* and cancel remaining sinks.
V ← V ∪ {t*}
forall sinks t ∈ V \ {t*} do
    E ← E ∪ {(t, t*)}
// Fix normal form of interval assignment.
I' ← run Algorithm 3.1 on G and I
if ∃v ∈ V : I'(v) = ∅ then
    return NO EMBEDDING
Γ' ← interval planar embedding of G under I' (as in Section 3.2)
return Γ' \ ({t*} ∪ {added dummy edges})
```

3.5.2. General Embeddings

Now that we know how to embed single source graphs under a fixed combinatorial embedding we are going to see, that this cannot efficiently be extended to general embeddings. To prove NP-completeness we are going to show a reduction from an NP-complete scheduling problem.

Definition 3.18 Sequencing With Release Times And Deadlines

Instance Given a set of tasks $T = \{t_1, \ldots, t_n\}$ with individual release times $r_1, \ldots, r_n \in \mathbb{N}$, deadlines $d_1, \ldots, d_n \in \mathbb{N}$ and processing times $p_1, \ldots, p_n \in \mathbb{N}$ for each task.

Question Is there a non-preemptive one-processor schedule for T, such that each task is scheduled after its release time and finished before its deadline? Formally, is there a function $\sigma : T \to \mathbb{N}$, such that for each $i \in \{1, \ldots, n\}$ we get:

- $\sigma(t_i) \geq r_i$, so each task starts after its release time.
- $\sigma(t_i) + p_i - 1 \leq d_i$, so each task finishes before its deadline.
- $\sigma(t_i) > \sigma(t_j) \Rightarrow \sigma(t_i) \geq \sigma(t_j) + p_j$ for any $j \in \{1, \ldots, n\} \setminus \{i\}$, so no two tasks are executed at the same time.

Garey and Johnson [6] showed that Sequencing With Release Times And Deadlines is strongly NP-complete in 1977: It is NP-complete, even if the sum of the processing times $\sum_{i=1}^{n} p_i$ is bounded by a polynomial in n. This is a very important property, because in the given reduction this will guarantee that the resulting instance of Interval Planarity will have polynomial size.

Theorem 3.19 Interval Planarity *is NP-complete, even if the given graph is a single source graph.*

Proof We already know that Interval Planarity is in NP from Theorem 3.11. Testing a given embedding for interval planarity can be done in polynomial time.

Let $T = \{t_1, \ldots, t_n\}$, $r_1, \ldots, r_n$, $d_1, \ldots, d_n$ and $p_1, \ldots, p_n$ be an instance of Sequencing With Release Times And Deadlines with $\sum_{i=1}^{n} p$ bounded by a polynomial in n. We will give a polynomial transformation into an Interval Planarity instance with a single source. Start with $G = (V, E)$ to be the base gadget shown in Figure 3.8a. Additionally for each task $t_i \in T$ add one of the task gadgets shown in Figure 3.8b. The base and all the task gadgets contain two common vertices u and v at which they are merged. The task gadget for job i consists of two vertices u, v and a path $P_i = \{x_1, \ldots, x_{p_i}\}$ above those two, so the number of vertices on P_i equals the processing time p_i of task t_i. Each vertex on P_i gets interval $[r_i, d_i]$, which corresponds to all possible time slots this task can be executed at. An example of G transformed from two tasks is shown in Figure 3.8c. In this example one of the tasks had length 3, the other length 4. The special vertices u, v and s all get interval $(-\infty, 0]$.

We claim that G is interval planar, if and only if there is a valid one-processor schedule for the Sequencing With Release Times And Deadlines instance.

Start with a valid schedule σ. Since σ is non-preemptive, it induces a total order on the tasks, without loss of generality $\sigma(t_1) < \ldots < \sigma(t_n)$. Order the outgoing edged to the task gadgets at u (and therefore symmetrically at v) from t_n to t_1 from left to right. Then in a planar embedding the gadget corresponding to t_n is incident to the outer face. The gadget for t_{n-1} is just inside of it, itself containing the remaining gadgets until the gadget for t_1 is the innermost one. For any $i \in \{1, \ldots, n-1\}$ we know that $\sigma(t_i) + p_i \leq \sigma(t_{i+1})$. Thus we

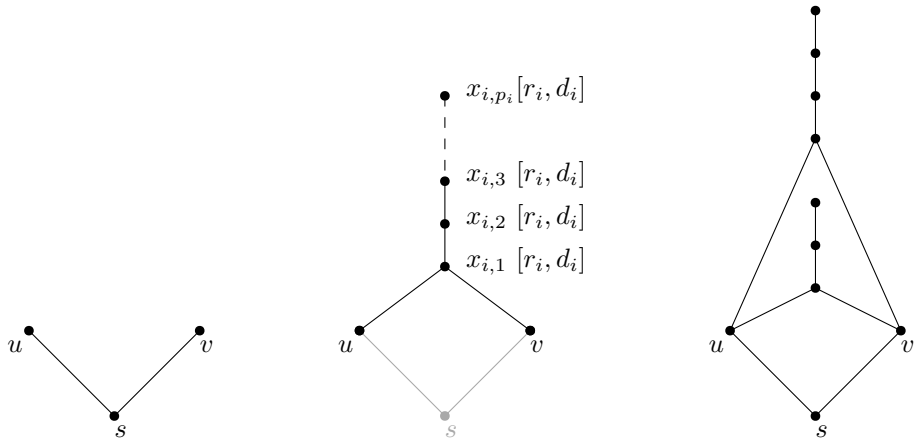

Figure 3.8.: The gadgets for the reduction to prove the NP-completeness of Interval Planarity for graphs with only a single source. (a) The base gadget. (b) One of these gadgets is needed for every task. The gray part illustrates how it is merged with the base gadget. (c) The base and all task gadgets are merged at their common vertices u and v.

.

can embed the j-th vertex of the path of t_i's gadget at y-coordinate $\sigma(t_i) + j - 1$ to get an interval planar embedding. All vertices get x-coordinate 0, except for u and v, which get -1 and 1 respectively.

Now consider an interval planar embedding of G. For the i-th task gadget define y_i to be the y-coordinate of $x_{i,1}$, the lowest vertex in the path P_i of the gadget. Then set $\sigma(t_i) = y_i$. Since the path of the i-th gadget has length p_i, the lowest vertex of any gadget above it must be at least at y-coordinate $y_i + p_i$. Therefore for all $t_j > t_i$ we get $\sigma(t_j) \geq \sigma(t_i) + p_i$, so we have a valid schedule. ■

3.6. Trees and Outerplanar Graphs

Every tree is upward planar [4] and for outerplanar graphs, Papakostas [13] showed that upward planarity can be tested in polynomial time. We will show that testing for interval planarity of trees is NP-complete. Then the NP-completeness for outerplanar graphs follows immediately. We can use a reduction very similar to the one used for single source graphs in Section 3.5.2.

Theorem 3.20 Interval Planarity *is NP-complete, even if the given graph $G = (V, E)$ is an oriented tree.*

Proof We already know that Interval Planarity is in NP, because a given embedding can be tested for interval planarity in polynomial time.

Again, let $T = \{t_1, \ldots, t_n\}$, $r_1, \ldots, r_n$, $d_1, \ldots, d_n$ and $p_1, \ldots, p_n$ be an instance of Sequencing With Release Times And Deadlines with $\sum_{i=1}^{n} p_i$ bounded by a polynomial in n. We are going to transform this into an instance of Interval Planarity where the graph G is a tree in polynomial time.

We start with the base gadget shown in Figure 3.9a. It consists of vertices u and v with interval $[0, 0]$ and vertices $c_1, \ldots, c_k$, where k is a constant bigger than the greatest deadline. Vertex u will be the connection to all other gadgets that are going to be added.

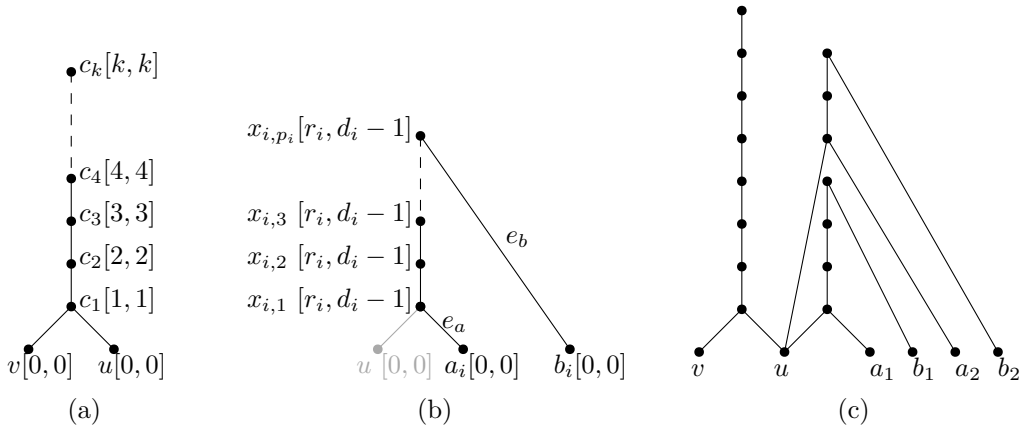

Figure 3.9.: Gadgets needed to show NP-completeness of Interval Planarity on trees. (a) The base gadget that the task gadgets get connected to. (b) One of these gadgets is needed for every task. The gray part does not belong to the gadget but shows how u is connected with it. (c) All task gadgets are merged at the common vertex u. In this case there are two tasks with processing times 3 and 4 respectively.

Now we add one gadget as shown in Figure 3.9b for each task $t_i \in T$. It consists of two base level vertices a_i and b_i as well as a path $P_i = \{x_{i,1}, \ldots, x_{i,p_i}\}$ of p_i vertices above them, each getting interval $[r_i, d_i]$. Vertex $x_{i,1}$, the lowest vertex of path P_i, gets connected to u. If there are n task gadgets, u has $n+1$ outgoing edges, the leftmost of which belongs to the base gadget. This is because each task gadget contains a vertex at level 1 or higher and thus cannot be embedded below the triangle-top formed by $v \to c_1 \leftarrow u$. So we know that all task gadgets are on the same side of the long path of vertices $c_1, \ldots, c_k$ and all vertices a_i and b_i are on the same side of vertex u on level 0. An example with two tasks with processing times 3 and 4 is shown in Figure 3.9c. The resulting graph G is a tree, as none of the gadgets has a cycle and two gadgets intersect at exactly one vertex. We claim that G has an interval planar embedding, if and only if there is a valid schedule for the Sequencing With Release Times And Deadlines instance.

Let σ be a valid schedule. Again, σ induces a total order on the tasks and we order the outgoing edges to task gadgets at vertex u according to σ from left to right (edge (u, c_1) still being the leftmost outgoing edge). We can embed the vertex $x_{i,k}$ of the i-th gadget at position $\sigma(t_i) + k - 1$ to get an interval planar embedding.

Now let Γ be an interval planar embedding, we need to extract a schedule for the tasks $t_1, \ldots, t_n$. We claim that the order of the outgoing edges to the task gadgets around u from left to right gives a valid schedule (each edge belongs to exactly one gadget). We need to show that in each interval planar embedding of G the following holds: If some vertex $x_{i,k}$ of the i-th gadget is below a vertex $x_{j,l}$ of the j-th gadget, all vertices $x_{i,1}, \ldots, x_{i,p_i}$ of the i-th gadget are below all vertices $x_{j,1}, \ldots, x_{j,p_j}$ of the j-th gadget.
Assume that this condition is violated, so there is some vertex $x_{j,l}$ that is embedded above $x_{i,1}$ and below x_{i,p_i} and without loss of generality the i-th gadget is left of the j-th gadget (by the order they are connected to vertex u). See Figure 3.10 for an illustration of this situation. The key observation here is that there is an $u \to x_{j,l}$ path P. But in order to get from u to $x_{j,l}$, path P needs to go around vertex a_i on level 0 and thus initially downward from u. This is impossible, because we required each edge and therefore each path to be y-monotone. Thus the order of the task gadget at vertex u indeed induces a total order on the tasks that fulfills the requirements of a schedule. ∎

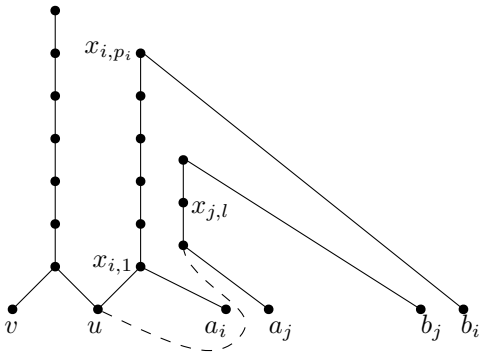

Figure 3.10.: $x_{j,l}$ lies between $x_{i,1}$ and x_{i,p_i}. But there must be a y-monotone path from $x_{j,l}$ to u, which is impossible as it would have to go "around" a_i (see the dashed edge, which is not monotone).

Corollary 3.21 Interval Planarity *is NP-complete, even if the given graph $G = (V, E)$ is an outerplanar graph.*

Proof Every tree is outerplanar, because any embedding of a tree has exactly one face. All vertices $v \in V$ lie on the boundary of this face. We immediately get from Theorem 3.20 that Interval Planarity remains NP-complete, when the graph G is outerplanar. ∎

3.7. Cycles

As a last graph class we look at the interval planarity of simple cycles and develop an algorithm to decide whether they are interval planar. We are then providing some ideas how to generalize this approach to embed cycles that are extended by attached trees or similar subgraphs.

In a cycle every vertex has degree 2, so no matter how we embed the adjacent edges at any vertex, the cyclic order around it is always the same. This gives us a lot of freedom when looking for an embedding of the cycle. If we have an upward planar embedding Γ, we call it *convex*, if each vertical line intersects the interior of the cycle at most once. An example for a convex embedding of a cycle can be seen in Figure 3.11a, while the embedding in Figure 3.11b is not convex.

Conjecture 3.22 *Let $G = (V, E)$ be a cycle and let I be an interval assignment. If Γ is an interval planar embedding of G under I, then a convex interval planar embedding Γ_c of G under I exists.*

We will not prove Conjecture 3.22 in this thesis but we will provide some ideas how a proof could look like in Appendix A. Based on this conjecture we will only construct convex embeddings in this chapter. We assume that no interval planar embedding exists at all, if we are not able to find a convex one. In any convex upward planar embedding we can uniquely distinguish between two types of sources and sinks:

Definition 3.23 Let $G = (V, E)$ be a cycle and Γ be a convex and upward planar embedding. Further let $W = (v_1, \ldots, v_{|V|})$ be a circular walk around the cycle in counterclockwise direction. Since Γ is convex, we can partition W into a part $W_{l\to r}$ where we only walk from left to right and a part $W_{r\to l}$ where we only walk from right to left. We get $W_{l\to r} \dot\cup W_{r\to l} = W$. The sources and sinks on $W_{l\to r}$ are called *lower sources* and *lower sinks*. On the other hand, the sources and sinks on $W_{r\to l}$ are called *upper sources* and *upper sinks*.

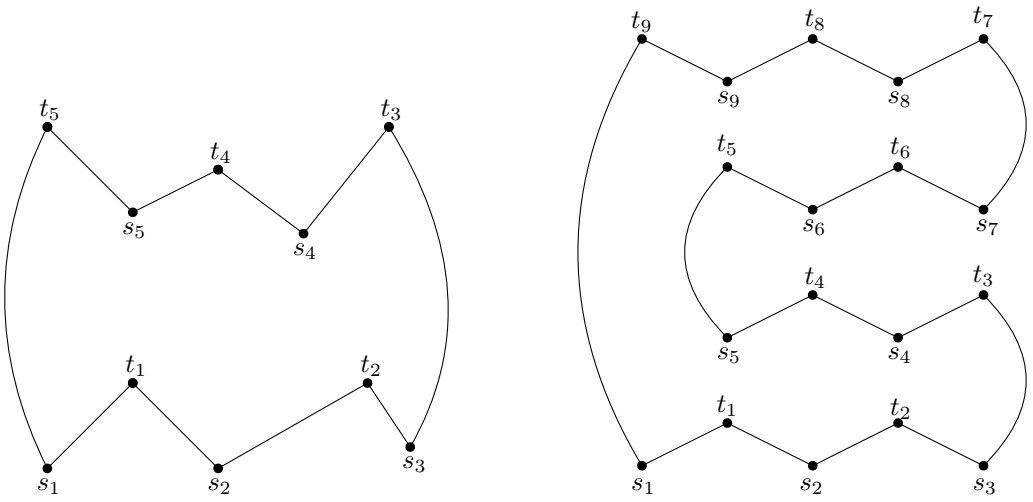

Figure 3.11.: (a) Any vertical line intersects the cycle at most once in this embedding. (b) A vertical line going through s_2, t_4, s_6 and t_8 would intersect the cycle twice.

In Figure 3.11a the sources s_1, s_2 and s_3 are lower sources while s_4 and s_5 are upper sources. Similarly, t_1 and t_2 are lower sinks while t_3, t_4 and t_5 are upper sinks. Again, our strategy to find an interval planar embedding is to augment the cycle into an st-planar graph. To do this we need to cancel all lower sources with upper sources and all lower sinks with upper sinks by inserting dummy edges. The lower sources and upper sinks can then be canceled with a supersource and a supersink, so that we can use the known methods from Section 3.2.

Lemma 3.24 *Let $G = (V, E)$ be a cycle, I be an interval assignment in normal form, s be a lower source and S be an upper source in some upward planar embedding of G. We can determine whether s can be canceled with S in $\mathcal{O}(1)$ for an interval planar embedding.*

Proof To verify that s can be canceled with S by adding a dummy edge between them, we need to show that all intervals in I remain non-empty after computing a new normal form with the added dummy edge. The important observation is that no matter how we place dummy edges in G, we will never get a directed path in G with more than one dummy edge on it. This is because a dummy edge is always oriented from a lower source (or sink) to an upper one and there is in general no edges from upper vertices to lower vertices. Because of this, we have no transitive dependencies between different dummy edges and it is enough to verify that the intervals of the endpoints of the dummy edges remain consistent after adding the dummy edge. Remember that we called the intervals of s and S consistent, if there are some $l_s \in I(s)$ and $l_S \in I(S)$ with $l_s < l_S$. Because I is in normal form, all vertices precedding s can then be embedded below l_s while all vertices following S can be embedded above l_S. ∎

Lemma 3.24 can equally be stated for lower and upper sinks and we will use it equally for sources and sinks in the following sections.

3.7.1. Cycle Operations

A combinatorial embedding of a cycle does not yet define which sources (or sinks) are the upper and which are the lower ones. Actually we can label the sources and sinks as upper (or lower) sources and sinks arbitrarily as long as all upper sources and upper sinks appear

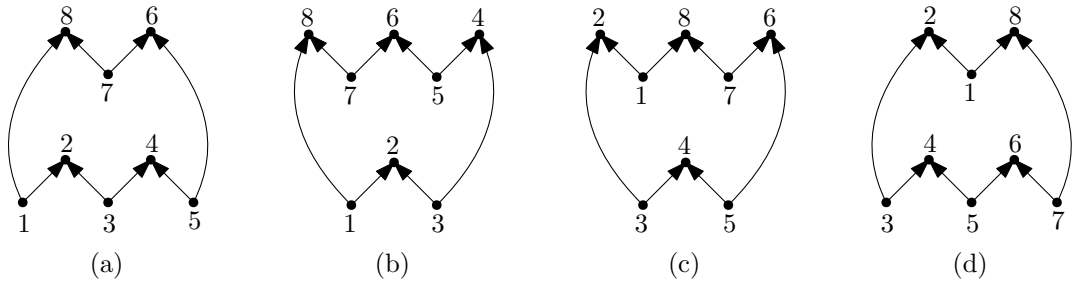

Figure 3.12.: (a) A sample cycle. (b) The cycle from 3.12a after a right flip. (c) The cycle from 3.12a after a left flip. (d) The cycle from 3.12a after a shift.

consecutive on the cycle. Then the lower sources and lower sinks automatically form a consecutive sequence as well. Of course there needs to be at least one lower source and at least one upper sink. In Figure 3.12 the same graph is shown four times with different upper and lower sources and sinks.

Lemma 3.25 *Let $G = (V, E)$ be a cycle. Between any two sources is a sink and between any two sinks is a source.*

Proof Let s_1 and s_2 be two consecutive sources and let $P = \{s_1 = v_1, \ldots, v_k = s_2\}$ be a path between them. Assume there is no sink on P. Then all inner vertices on P have exactly one incoming and one outgoing edge (otherwise they would either be a source or a sink). Since s_1 is a source, edge (v_1, v_2) is oriented towards v_2. Then edge (v_2, v_3) must be oriented towards v_3. Inductively we get that edge (v_{k-1}, v_k) must be oriented towards $v_k = s_2$. But this is a contradiction, because s_2 is a source and thus has no incoming edges. There must have been a sink on P. By the same argument we can show that there is a source between any two sinks. ∎

Definition 3.26 Let $G = (V, E)$ be a cycle with a convex upward planar embedding Γ.

- In a *right flip* $\mathcal{R}(\Gamma)$, the rightmost lower source in Γ becomes the rightmost upper source and the rightmost lower sink in Γ becomes the rightmost upper sink. Figure 3.12b shows the graph from Figure 3.12a after one right flip.
- In a *left flip* $\mathcal{L}(\Gamma)$, the leftmost lower source in Γ becomes the leftmost upper source and the leftmost lower sink in Γ becomes the leftmost upper sink. Figure 3.12c shows the graph from Figure 3.12a after one left flip.

Note that both operations can be inversed: We will denote this with $\mathcal{R}^{-1}(\Gamma)$ and $\mathcal{L}^{-1}(\Gamma)$. A flip can only be applied, if there are at least two lower sources in Γ (and by Lemma 3.25 a lower sink between them). Similarly, an inverse flip can only be applied, if there is at least one upper source in Γ.

Definition 3.27 Let $G = (V, E)$ be a cycle with a convex upward planar embedding Γ. A *shift* $\mathcal{S}(\Gamma)$ is a combination of a left flip and an inverse right flip executed in any order. Figure 3.12d shows the graph from Figure 3.12a after one shift.

Note that a shift is always well defined, even in the degenerated cases with just one lower source or just one upper sink. If there is exactly one lower source and exactly one upper sink, a shift has no effect. If there is at least two of either kind, either $\mathcal{L}$ or $\mathcal{R}^{-1}$ is defined and the other can then be executed afterwards.

Shifts and flips allow us to enumerate all possible assignments of which sources and sinks are lower ones and which are upper ones. If the cycle contains S sources, we can have any

number between 1 and S of lower sources. To get from an assignment with k lower sources to one with $k-1$, apply a left (or right) flip. To get to an assignment with $k+1$ lower sources apply a left (or right) inverse flip. By shifting, we can enumerate all assignments with the same number of lower sources, because a single shift does not change the number of lower sources.

3.7.2. Fixed Assignment of Lower and Upper Sources and Sinks

Let $G = (V, E)$ be a cycle, let Γ be a convex upward planar embedding and let I be an interval assignment in normal form. Note that the convex upward planar embedding Γ already defines which sources and sink are lower and which are upper ones. In this section we will describe an algorithm to embed G. We need to cancel all lower sources to upper sources and all lower sinks to upper sinks. However we might have several choices which lower source (or sink) we can cancel with which upper source (or sink) and not every choice will allow an interval planar embedding, i.e. result in an interval assignment in normal form with non-empty intervals. We will describe an $\mathcal{O}(|V|^3)$ dynamic programming algorithm that finds a valid embedding or detects that none exists. In Section 3.7.3 we are going to apply this algorithm to any possible assignment of lower and upper sources and sinks.

The convex upward planar embedding Γ gives us a left to right order on the lower sources and sinks as well as a left to right order on the upper sources and sinks. Between any two sources lies a sink and between any two sinks lies a source by Lemma 3.25. Therefore, if our cycle has k lower sources and l upper sinks, it has $k-1$ lower sinks and $l-1$ upper sources. Algorithm 3.5 processes the lower sinks from left to right. For a lower sink t, we identify all upper sinks T that it can be canceled with, such that all lower sinks left of t and all upper sources left of T can be canceled as well. If and only if there is such an upper sink T for the rightmost lower sink t and all upper sources to the right of T can be canceled with the rightmost lower source, then an interval planar embedding exists and can be constructed.

In Algorithm 3.5 the lower sources and sinks are denoted with lowercase letters s and t, the upper sources and sinks with uppercase letters S and T. We are now going to look at the different steps of the algorithm (as they are labeled in the given pseudocode). Whenever we check whether a lower source (or sink) can be canceled with an upper one we do so as in Lemma 3.24.

Initialization. In Line 5 we initialize a two-dimensional $k-1 \times l$-array `possible`, where $k-1$ is the number of lower sinks and l is the number of upper sinks. If an entry `possible[i][j]` equals `true`, this means that lower sink t_i can be canceled with upper sink T_j and that all upper sources and lower sinks to the left of either of these can also be canceled. This step needs $\mathcal{O}(|V|^2)$ time, since both k and l are in $\mathcal{O}(|V|)$.

Base case. We find the possible upper sinks to cancel lower sink t_1 with in lines 6-9 by iterating over all upper sinks. For a candidate T_j we further need to check, if we can cancel all upper sources left of T_j (these are $S_1, \ldots, S_{j-1}$) with lower source s_1, since this is the only lower source left of t_1. We need $\mathcal{O}(|V|)$ time to do this, if we do not recheck each upper source in each iteration in line 8 but only once when it first is left of the current upper sink T_j.

Recursive case. We now deal with the remaining lower sinks in lines 10-15. For a lower sink t_i we find all possible upper sinks T_j that we can cancel t_i with. To guarantee that everything left of t_i and T_j can be canceled as well, we recursively ask for the biggest index $j' \leq j$, such that t_{i-1} can be canceled with $T_{j'}$ and everything left of it can be canceled. What remains to check are all upper sources between $T_{j'}$ and T_j (these are $S_{j'}, \ldots, S_{j-1}$). We need to test, that we can cancel them with s_i, which is the only lower source between t_i

Algorithm 3.5: Embed Cycle with Assignment of Lower and Upper Sources/Sinks

Input: Directed Cycle $G = (V, E)$, upward planar embedding Γ, interval assignment I in normal form, assignment of lower and upper sources/sinks.
Output: Γ' interval planar embedding of G or `NO EMBEDDING`, if none exists.

```
// Initialization.
1  {s_1, ..., s_k} ← lower sources (left to right)
2  {S_1, ..., S_{l-1}} ← upper sources (left to right)
3  {t_1, ..., t_{k-1}} ← lower sinks (left to right)
4  {T_1, ..., T_l} ← upper sinks (left to right)
5  possible[1..k-1][1..l] ← false
// Base case: First lower sink.
6  for j = 1 to l do
7      if t_1 can be canceled with upper sink T_j then
8          if upper sources S_1, ..., S_{j-1} can be canceled with lower source s_1 then
9              possible[1][j] ← true
// Recursive case: Remaining lower sinks.
10 for i = 2 to k-1 do
11     for j = 1 to l do
12         if lower sink t_i can be canceled with upper sink T_j then
13             foreach j' ≤ j with possible[i-1][j'] = true do
14                 if sources between T_{j'} and T_j can be canceled with source s_i then
15                     possible[i][j] ← true
// Check existence.
16 if ∀j ∈ {1, ..., l} : possible[k-1][j] = false then
17     return NO EMBEDDING
18 j' ← arg max_{j∈{1,...,l}} possible[k-1][j] = true
19 if some source S ∈ {S_{j'}, ..., S_{l-1}} cannot be canceled with lower source s_k then
20     return NO EMBEDDING
// Ebedding exists: Now cancel sources and sinks.
21 E ← E ∪ {(s_k, S_{j'}), ..., (s_k, S_{l-1})}
22 pos ← l
23 for i ← k-1 to 2 do
24     j' ← arg max_{j∈{1,...,pos}} possible[i][j] = true
25     j'' ← arg max_{j∈{1,...,j'}} possible[i-1][j] = true
26     E ← E ∪ {(t_i, T_{j'})}
27     E ← E ∪ {(s_i, S_{j''}), ..., (s_i, S_{j'-1})}
28     pos ← j''
29 E ← E ∪ {(t_1, T_pos)}
30 E ← E ∪ {(s_1, S_1), ..., (s_1, S_{pos-1})}
// Add supersource s and supersink t. Then embed st-planar graph.
31 V ← V ∪ {s, t}
32 E ← E ∪ {(s, s_1), ..., (s, s_k), (T_1, t), ..., (T_l, t)}
33 I(s), I(t) ← [-∞, ∞]
34 Γ' ← interval planar embedding of G (Section 3.2)
35 return Γ' \ {added dummy edges and vertices s, t}
```

and t_{i-1}. This step of the algorithm takes $\mathcal{O}(|V|^3)$ time. The outer loop iterates over $\mathcal{O}(|V|)$ lower sinks. Each can be processed in $\mathcal{O}(|V|^2)$ as in the base case.

Existence check. In lines 16-20 we check whether an interval planar embedding of the cycle exists. First we check whether the rightmost lower sink t_{k-1} can be canceled with some upper sink. If so, let $T_{j'}$ be the rightmost of it. If not, there is of course no interval planar embedding. But even if that is possible (and therefore everything left of it is as well), we still need to check all upper sources right of $T_{j'}$ (these are $S_{j'}, \dots, S_{l-1}$). They all need to be canceled with lower source s_k (the only one right of t_{k-1}). This step can be done in $\mathcal{O}(|V|)$, since we just need to iterate over one dimension of the `possible`-array and afterwards once over the upper sources.

Source and sink canceling. The information from the `possible` array is used in lines 21-30 to find a valid assignment of lower sources and sinks to the upper sources and sinks. We know from the previous step, that $S_{j'}, \dots, S_{l-1}$ can be canceled with s_k. In line 21 these dummy edges are added. We then cancel all upper sources and sinks from right to left. Variable `pos` always marks the rightmost upper sink, that we can still cancel lower sinks to, initially this is T_l. In each step we connect the current rightmost lower sink t_i to the upper sink that is as far right as possible but left of or equal to T_{pos}, we call this one $T_{j'}$. The corresponding dummy edge is added in line 26. We also query where t_{i-1} will be connected to, and call this upper sink $T_{j''}$. All upper sources between $T_{j''}$ and $T_{j'}$ are canceled with the unique lower source s_i between t_{i-1} and t_i in line 27. After the loop we still need to take care of the leftmost lower sink t_1 and all upper sources left of the upper sink that t_1 is canceled with. The construction is done in $\mathcal{O}(|V|^2)$ by a traversal of the `possible`-array in the reverse order it was computed.

Embedding. Lines 31-35 now take the graph without any upper sources and lower sinks and extend it by a supersource s, a supersink t and dummy edges from s to all lower sources and from all upper sinks to t. Then the known algorithm to embed st-planar graphs from Section 3.2 can be used to find an interval planar embedding. Removing the supersource and the supersink as well as all added dummy edges then gives an interval planar embedding of the cycle. The embedding can be done in $\mathcal{O}(|V|)$. Only $\mathcal{O}(|V|)$ edges are added (and afterwards removed) and the known technique to embed st-planar graphs also takes $\mathcal{O}(|V|)$ time.

In total the runtime of Algorithm 3.5 is in $\mathcal{O}(|V|^3)$, dominated by the recursive step to fill the `possible`-array. This array itself dominates the space requirement, which is in $\mathcal{O}(|V|^2)$.

3.7.3. General Cycle Embedding

We are now going to combine the cycle operations from Section 3.7.1 and the embedding algorithm from Section 3.7.2 to an algorithm that can find an interval planar embedding for any given cycle or detect that no such embedding exists.

Lemma 3.28 *Let $G = (V, E)$ be a cycle. G is upward planar, if and only if G contains at least one source and at least one sink. Further an upward planar embedding of G with exactly one lower source exists. All other sources and sinks are upper sinks.*

Proof Assume that G contains no sources or sinks. Then each vertex $v \in V$ has exactly one incoming and one outgoing edge, so we can walk around G following the direction of the edges. But then by transitivity every vertex $v \in V$ needs to be strictly below any other vertex $u \in V$ and even below itself in any upward planar embedding, which is impossible.

Now assume G has at least one source. We will construct an upward planar embedding Γ, assuming that G contains only sources and sinks as vertices. Any other vertices with exactly one incoming and one outgoing edge can then be added to the upward planar

Algorithm 3.6: Embed Cycle

Input: Directed Cycle $G = (V, E)$, interval assignment I in normal form.
Output: Γ' interval planar embedding of G or `NO EMBEDDING`, if none exists.

```
   // Guarantee that the cycle is upward planar.  (Lemma 3.28)
 1 if G does not contain sources or sinks then
 2     return NO EMBEDDING

   // Initialization.
 3 S ← number of sources of G
 4 Γ ← upward planar embedding of G with exactly one lower source (Lemma 3.28)

   // We have between 1 and S lower sources.
 5 for i = 1 to S do
       // Enumerate assignments with exactly i lower sources.
 6     for j = 1 to S do
 7         Γ' ← interval planar embedding from Algorithm 3.5
 8         if Γ' ≠ NO EMBEDDING then
 9             return Γ'

           // Shift to get to the next assignment with i lower sources.
10         Γ ← S(Γ)
       // Inverse right flip to get i + 1 lower sources.
11     Γ ← R^{-1}(Γ)    (Ignore if all sources are lower sources and this is not defined.)

12 return NO EMBEDDING
```

embedding later. All vertices on a directed path P from a source s to a sink t can be placed arbitrarily on the edge between s and t in the constructed upward planar embedding as long as they appear in the same order as in P.

Let $s \in V$ be an arbitrary source of G. It will be the only lower source in Γ. Initially place s at position $(0, 0)$ and all other vertices in the order they appear on the cycle from left to right on consecutive x-coordinates and at y-coordinate 2. Every source $s' \in V \setminus \{s\}$ is now pulled down to y-coordinate 1. For the given coordinates, edges can easily be added to obtain an upward planar drawing. ∎

Pseudocode to embed a cycle is given in Algorithm 3.6. The idea is to systematically enumerate all possible assignments of which sources and sinks are lower ones and which are upper ones. It starts with an upward planar embedding Γ that has a single lower source, which exists by Lemma 3.28. Then S shift operations are applied, where S is the number of sources of G. This way, each possible assignment of which sources and sinks are lower ones and which are upper ones with exactly one lower source is generated once. Each of them is tested for an interval planar embedding with Algorithm 3.5 in Line 7. If no interval planar embedding is found, an inverse right flip is applied to Γ in Line 11. This transforms Γ into an upward planar embedding with exactly two lower sources and one lower sink. Again, S shifts are applied to test all assignments with two lower sources and each is tested for an interval planar embedding. This process continues until all S sources are lower sources. In total every possible assignment of which sources and sinks are lower ones and which are upper ones is tested once.

Theorem 3.29 *Let $G = (V, E)$ be a cycle and I be an interval assignment in normal form. We can find a convex interval planar embedding of G under I in $\mathcal{O}(|V|^5)$ time or detect that none exists.*

Proof Flips, inverse flips and shifts can all be applied to an upward planar embedding Γ in constant time. Algorithm 3.6 iterates over the number of sources S which is in $\mathcal{O}(|V|)$. In each iteration it applies S shifts, so the total number of calls of Algorithm 3.5 is in $\mathcal{O}(|V|^2)$. Each of these calls takes time in $\mathcal{O}(|V|^3)$, so in total we get a time in $\mathcal{O}(|V|^5)$. Since every possible assignment of which sources and sinks are lower ones and which are upper ones is tried once, we definitely find a convex embedding if one exists. ∎

3.7.4. Generalizations of Cycle Embedding

In this section we collect some ideas how the developed algorithm to embed a cycle can be extended to graphs that are a little bit more complex.

- Consider cycles that already have some chords connecting sources with other sources or sinks with other sinks. If the resulting graph is still planar, these chords then partition the single inner face of the cycle into several ones, each of them might contain several sources and sinks. We can only handle cases where chords start at lower sources (or sinks) and end at upper ones. Therefore the chords already fix some sources (and sinks) to be lower ones and some to be upper ones. All sources and sinks between two lower sources (or sinks) are then also lower ones. Similarly, all sources and sinks between two upper sources (or sinks) are upper ones as well. This fixes most of the sources and sinks to be lower or upper ones. Only the sources and sinks left of the endpoints of the leftmost chord or right of the endpoints of the rightmost chord could still be chosen to be lower or upper ones. Again, executing left or right (inverse) flips can enumerate all possible assignments. By using Algorithm 3.6 on each face independently, we can then find an embedding for the cycle with chords, if it exists.
- Consider cycles where some vertices are the single sources (or single sinks) of attached single source (or single sink) graphs. An example for such a graph is shown in Figure 3.13a. Single source and single sink graphs include single source/single sink trees, paths and st-planar graphs. The important property of these attached graphs is that their single source (or sink) is the only vertex that they share with the boundary of the cycle. How these attached subgraphs are embedded is not part of this section. For some types of graphs the known methods from earlier sections can be used.

  To extend Algorithm 3.6 we can make the following observations. Without loss of generality we assume that the attached subgraphs are single source graphs with several sinks.

  1. Attached subgraphs that are outside of the cycle can be canceled with the added supersource or supersink. They do not need to be considered when embedding the cycle. If it is possible to flip and shift the cycle such that it allows an interval planar embedding and all attached subgraphs are outside of the cycle, then Algorithm 3.5 can be used.
  2. If an inner subgraph contains multiple sinks, we can cancel them all to the same upper sink of the cycle: If the sink that will be embedded at the highest level among all sinks can be canceled at a particular upper sink of the cycle, all other and therefore not higher sinks can be canceled at the same upper sink as well.
  3. If there are multiple inner single source subgraphs with the same source vertex on the cycle, we can cancel all sinks of these subgraphs at the same upper sink of the cycle. The argument is similar as in observation 2: If the subgraph with the highest sink can be canceled at some upper sink of the cycle, the sinks of the others are not higher and can therefore be canceled at the same upper sink.

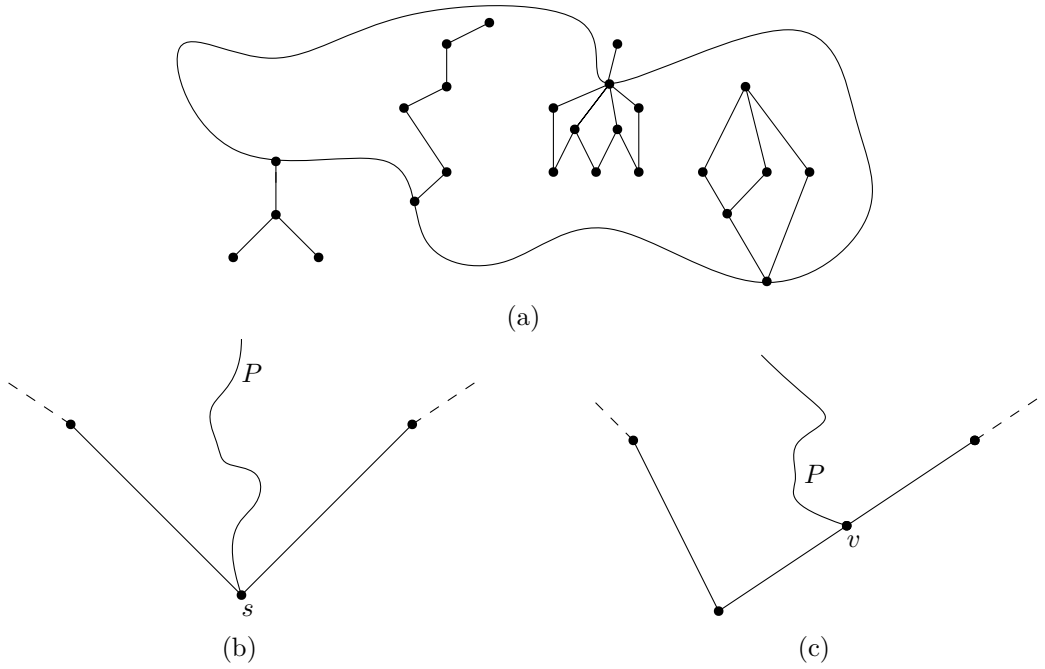

Figure 3.13.: (a) An example for a cycle with some inner graphs inside it. They are all single source or single sink graphs, so they can be embedded by the known methods. (b) The inner graph P is connected to the cycle at a lower source. In this case this gives two options how upper sources can be canceled at this source. (c) The inner graph P is connected to a vertex v that is neither a sink nor a source of the cycle. Upper sources can now be also canceled at v, even though it is not a lower source.

Observations 2 and 3 allow us, to treat all sinks of all subgraphs starting at the same vertex v on the cycle as a single sink of the graph when using Algorithm 3.6 to embed the cycle. But besides adding more sources and sinks, the attached subgraphs create additional options to cancel other sources and sinks to:

1. Figure 3.13b shows an inner subgraph P that is connected to the cycle at a lower source s of the cycle. For upper sources it now makes a difference whether they are canceled with s left of P or right of P. So in the list of lower sources we now need to duplicate s to s_{left} and s_{right} and always check both options to cancel upper sources. The same needs to be done for upper sinks t of the cycle that are the single sink of an inner subgraph.
2. Figure 3.13c shows an inner subgraph P that is connected to a vertex v on the cycle that is neither a source nor a sink of it. Therefore v would not have been a candidate to cancel upper sources with. However, since P starts at vertex v, it might be necessary to cancel an upper source with v "between" P and the boundary of the cycle.

3.8. Summary: Comparing Upward Planarity and Interval Planarity

Table 3.1 lists the different scenarios that we considered in this chapter. We first saw that both Upward Planarity and Interval Planarity are NP-complete on general graphs. We then looked at several restricted graph classes that allow for an efficient upward planarity test. In most cases Interval Planarity remains NP-complete. A notably exception are

	Upward Planarity	Interval Planarity (Any Embedding)	Interval Planarity (Fixed Embedding)
General Graphs	NP-complete[8]	NP-complete Theorem 3.11	NP-complete Theorem 3.13
st-planar Graphs	$\mathcal{O}(\|V\|)$[9]	$\mathcal{O}(\|V\|)$ Theorems 3.5, 3.8	
Single Source	$\mathcal{O}(\|V\|)$[1]	NP-complete Theorem 3.19	$\mathcal{O}(\|V\|)$ Theorem 3.17
Trees	$\mathcal{O}(1)$ All trees are upward planar. [4]	NP-complete Theorem 3.20	
Outerplanar	$\mathcal{O}(\|V\|^2)$[13]	NP-complete Corollary 3.21	
Cycles	$\mathcal{O}(\|V\|)$ Lemma 3.28	$\mathcal{O}(\|V\|^5)$ Theorem 3.29	

Table 3.1.: A comparison between Upward Planarity and Interval Planarity summarizing the results of this chapter.

st-planar graphs for which we gave fast embedding algorithms, even if the combinatorial embedding is not fixed. Also surprising are cycles: We can gave an embedding algorithm, but even if it runs in polynomial time, it is quite slow (and relies on Conjecture 3.22). Level Planarity can be tested in $\mathcal{O}(|V|)$ for any graph $G = (V, E)$, so it is not listed explicitly in the given table.

4. Interval Planar Embeddings with Bounded Width

In the previous chapter we looked at the existence of interval planar embeddings and saw how to construct them for some graph classes. Now we want to consider embeddings where the maximum number of vertices on each level is bounded. We can then binary search for the minimum number m, such that an interval planar embedding exists with where no level contains more than m vertices.

Definition 4.1 The *width* of an interval planar embedding Γ is the maximum number of vertices that share the same y-coordinate in Γ, i.e. are embedded in the same level.

This definition is motivated from scheduling problems, where a set of tasks needs to be scheduled on a bounded number of available machines. If each task has an individual release time and deadline, there is a precedence relation between the tasks and all tasks have unit processing time, the only difference to Interval Planarity is the planarity constraint. The idea is to use known scheduling methods for the level assignment and then create planar drawings of the graph with respect to these levels. Given a valid schedule, it is a classic Level Planarity problem to find a planar embedding. We know, that this can be constructed, if it exists, in $\mathcal{O}(|V|)$[11].

Note that our definition of width does not take edges into account. This is because of the similarity to the scheduling problem. If we would be interested in nicely readable drawings we should consider the number of edges crossing a certain level as well in our width definition.

In this chapter we will see that the existence of a schedule with some width does not imply the existence of an interval planar embedding with the same width. Further, we will describe some graph classes in which every schedule can be used to construct an interval planar embedding with the same width. This will be useful to show the NP-completeness of finding an interval planar embedding with bounded width even for instance that otherwise allow an efficient interval planar embedding, for example st-planar graphs.

4.1. Finding Schedules

Definition 4.2 Multiprocessor Scheduling

Instance Given n unit length tasks $t_1, \ldots, t_n$, individual release times $r_1, \ldots, r_n$ and deadlines $d_1, \ldots, d_n$ and a precedence relation between the tasks. Further, integer m is the number of available machines.

Question Is there a non-preemptive m-processor schedule obeying the precedence relation and executing all tasks in their allowed interval?

Theorem 4.3 Multiprocessor Scheduling *is NP-complete.*

Proof Multiprocessor Scheduling is in NP, since a given schedule can easily be checked to obey all precedences, release times and deadlines.

Ullman [14] proved NP-completeness of the simpler version without release times and only one overall deadline D for all tasks with a reduction from 3SAT. An easier reduction from Clique was given by Lenstra [12] even for the case $D = 3$. We can transform such an instance into a Multiprocessor Scheduling instance by just defining $r_i := 0$ and $d_i := D$ for all $i \in \{1, \ldots, n\}$. Obviously there is a bijection between the valid schedules of both instances, so that Multiprocessor Scheduling is indeed NP-complete. ∎

There is however a few special cases in which a schedule can be found in polynomial time. If all release times r_i are 0 and the precedence relation forms an in-tree, a schedule can be found efficiently (in an in-tree precedence relation, no task has more than one direct successor)[7]. If there are exactly two processors, Garey and Johnson [6] describe an $\mathcal{O}(|V|^3)$ algorithm to find a schedule.

4.2. Planar Embedding of Schedules

Constructing a planar embedding of a given schedule is a Level Planarity instance and can be done in linear time, if such an embedding exists. However, the existence of a schedule with some width, does not guarantee the existence of a level planar embedding of this schedule, as can be seen in Figure 4.1.

4.2.1. Planar Embedding of Paths and Forests

Assume we are given a schedule for a set $\mathcal{P}$ of paths on m machines. Our goal is to embed this schedule with planar and y-monotone edges. We imagine a grid of machine slots of width m and height d, where d is the time that the last job is scheduled at. In this grid, each line corresponds to one time unit, time running from bottom to top, and each column corresponds to one machine. We now try to assign the jobs to machine slots so that the resulting embedding is planar with y-monotone edges and each job is in the line that corresponds to its execution time.

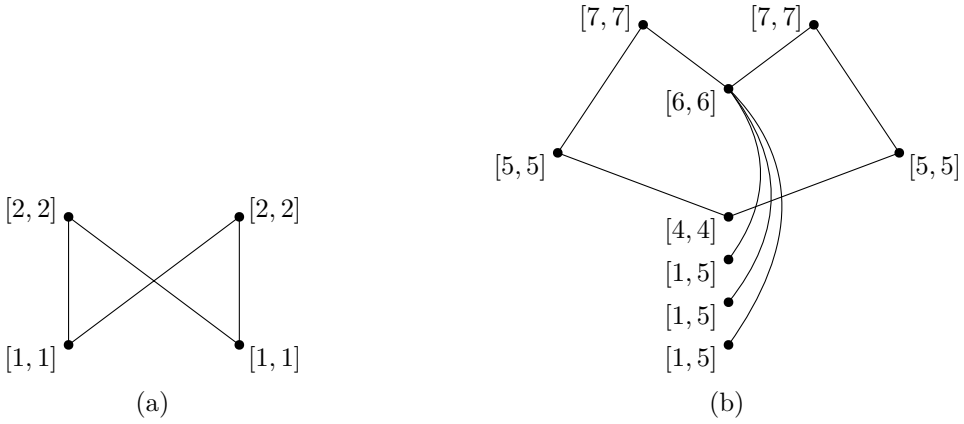

Figure 4.1.: (a) This instance can easily be scheduled, but there is no upward planar drawing of it. (b) A schedule of width 2 exists, but any planar drawing needs width 5. This example can be made arbitrarily worse by putting more vertices before the one with interval $[6, 6]$.

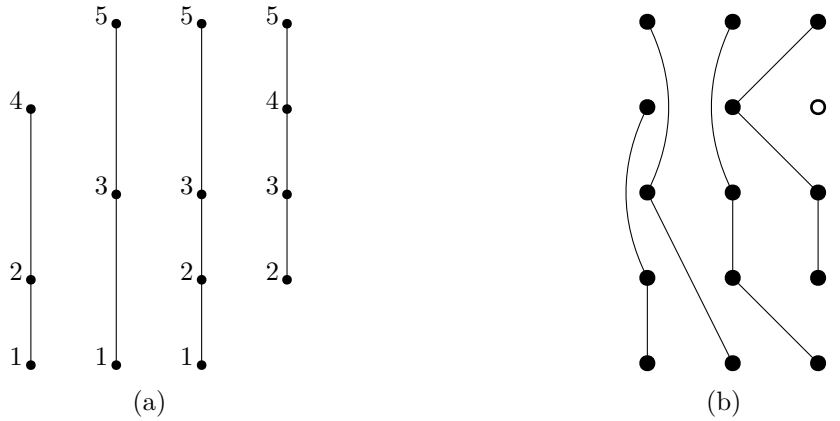

Figure 4.2.: (a) A schedule for four paths on three processors. (b) A planar embedding of the four paths as constructed in the proof for Theorem 4.4. The paths were embedded in left to right order. Each dot represents a processor slot at some time unit. The filled slots are occupied by tasks, the empty ones are free.

Theorem 4.4 *For a given set of paths $\mathcal{P} = \{P_1, \ldots, P_k\}$ scheduled on m machines it is always possible to embed them planar with y-monotone edges.*

Proof We can process the paths one by one in any order. Since we already have a schedule, we know on which line each vertex needs to be placed. We greedily choose the leftmost free slot on this line. This gives us an invariant that at every step every line has its first few (possibly zero) slots filled and all others right of that still free. We can therefore divide the grid at every step into a filled *left* subset and a free *right* subset. After placing the vertices of a path we can then draw the edges between them as y-monotone curves. On every level we draw them between the rightmost filled and the leftmost free slot. This construction is shown in Figure 4.2. ∎

We can extend this step by step construction to some special trees.

Definition 4.5 An *in-tree* is a tree in which no vertex has more than one direct successor. An *out-tree* is a tree in which no vertex has more than one direct predecessor.

Theorem 4.6 *Given a set of in-trees and out-trees $\mathcal{F}$ scheduled on m machines it is always possible to embed them planar with y-monotone edges.*

Proof To embed an out-tree, we start by getting an arbitrary upward planar embedding of it. We process the vertices in a left-first depth first search (DFS) order starting from the root. Place each vertex at the level it was scheduled at and all but the root are then connected to their DFS parent. When placing the vertices and drawing the edges we maintain the same invariant as in Theorem 4.4, meaning that we fill the slots on each level from left to right and always draw the edges between the rightmost filled and leftmost free slot on each level.

We process in-trees analogously, finding the leftmost paths from leaves to roots with a right first DFS in the reverse graph. An example of this embedding is given in Figure 4.3. ∎

We can further extend this results to trees that can be separated into an in-subtree and an out-subtree that share exactly one vertex (the root of both subtrees). Start by embedding the in-tree as in Theorem 4.6. In a second step the out-tree can be embedded, reusing the common root.

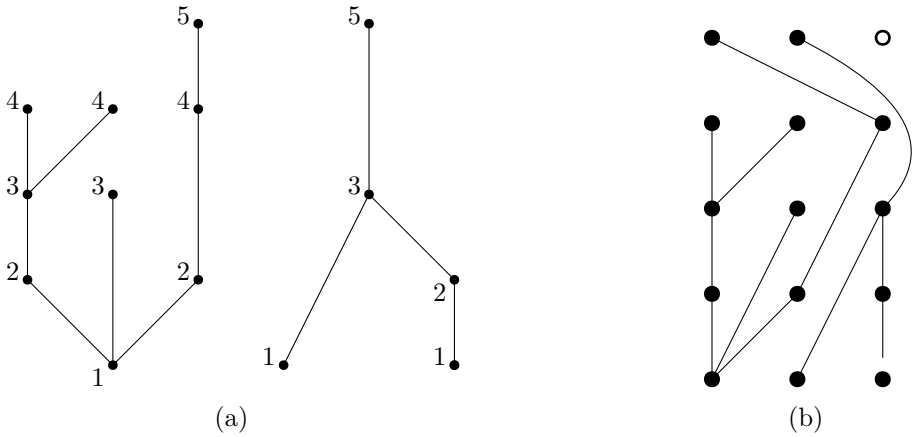

Figure 4.3.: (a) A schedule for two trees on three processors. (b) A planar embedding of the two trees as constructed in the proof for Theorem 4.6. The left tree was embedded first. Each dot represents a processor slot at some time unit. The filled slots are occupied by tasks, the empty ones are free.

4.3. NP-Completeness of Interval Planarity with a Bounded Width

We saw that computing a schedule for tasks with release times, deadlines and a precedence relation is NP-complete for a fixed number of available machines. We will use this to show that the same is true for interval planarity with a bounded width.

Definition 4.7 Bounded Width Interval Planarity

Instance Given a graph $G = (V, E)$, an interval assignment I and an integer m.

Question Is there an interval planar embedding of G under I with width not exceeding m?

Theorem 4.8 Bounded Width Interval Planarity *is NP-complete.*

Proof Bounded Width Interval Planarity is in NP. Given an embedding, we can check planarity in linear time. In constant time per vertex we can check, that each vertex was embedded at an integer y-coordinate inside its interval. By counting the number of vertices per level we can verify that no level contains more than m vertices.

To show that Bounded Width Interval Planarity is NP-complete, we need to show, that the restriction to upward planar graphs does not make Multiprocessor Scheduling easier. Garey and Johnson [7] showed that Multiprocessor Scheduling remains NP-complete, even if the precedence relation forms an out-tree. But every tree is upward planar and in Theorem 4.6 we saw, that we can always construct a planar embedding for schedules of out-trees without having to increase the width. So we get that Bounded Width Interval Planarity is NP-complete for out-trees and therefore also for general upward planar graphs. ∎

5. Conclusion

In this thesis we introduced Interval Planarity, a generalization of Upward Planarity and Level Planarity. The goal was to close the theoretical gap between these two well understood problems. Since any interval planar embedding is also upward planar, it was no surprise that Interval Planarity is NP-complete in its general case, because testing an embedding for upward planarity is also NP-complete.

We started by giving an explicit and efficient construction for interval planar embeddings of st-planar graphs in $\mathcal{O}(|V|)$. Based on this result we presented a combinatorial characterization of interval planar graphs: Every interval planar graph $G = (V, E)$ is a subgraph of an st-planar graph on the same vertex set and a supersource and a supersink, where the interval of each $v \in V$ is a subset of the interval assigned to v in G. This led to a general strategy to find interval planar embeddings: We need to cancel all sources and sinks except for the supersource and supersink while maintaining non-empty intervals in normal form. We successfully applied this strategy to single source graphs under a fixed combinatorial embedding, using the fact that all their sinks need to be canceled with a unique other vertex. An interval planar embedding can therefore be found in $\mathcal{O}(|V|)$. Another positive example are cycles, for which we described an $\mathcal{O}(|V|^5)$ embedding algorithm by enumerating possible upward planar embeddings. For each of them we then used dynamic programming to assign sources and sinks to others to augment the cycle to an st-planar graph.

Even though Upward Planarity can be efficiently tested for general single source graphs, trees and outerplanar graphs, we showed that Interval Planarity remains NP-complete in these cases. We further showed that fixing a combinatorial embedding (and even an outer face) is in general not enough to efficiently find an embedding. Interval Planarity remains NP-complete in this case.

In the last chapter we defined the width of an interval planar embedding to be the maximum number of vertices that share a level, i.e. are embedded with the same y-coordinate. Motivated by the problem to find a schedule for tasks with unit processing times, release times and deadlines on a minimum number of machines we tried to find interval planar embeddings of minimal width. We denoted this problem by Bounded Width Interval Planarity. We saw that Bounded Width Interval Planarity is NP-complete even for very simple graph classes like out-trees (trees with a single source). Bounded Width Interval Planarity is therefore NP-complete for even more graph classes than general Interval Planarity.

For further research, the following questions remain unanswered: First, is there an efficient algorithm to find an interval planar embedding for trees or outerplanar graphs both with a fixed combinatorial embedding (and maybe outer face)? We saw that both are NP-complete for general embeddings. The second regards Conjecture 3.22, that our $\mathcal{O}(|V|^5)$ cycle embedding algorithm relies on. We believe that every cycle that has an interval planar embedding also has a convex one. In this case convex means that any vertical line intersects the interior of the cycle at most once. We left this conjecture without a proof, however some thoughts an a proof outline are given in Appendix A. The idea is that a non convex embedding can iteratively be converted into a convex embedding while maintaining interval planarity in each executed step.

The goal of this thesis was to close the gap between Upward Planarity and Level Planarity by looking at a more general problem, that can describe instances "between" the two. Upward Planarity is NP-complete, Level Planarity can always be solved in linear time. What we found is that Interval Planarity is in many cases harder than Upward Planarity. Both are NP-complete for general graphs, but even for many restricted graph classes Interval Planarity is efficiently solvable.

Bibliography

[1] Paola Bertolazzi et al. "Optimal Upward Planarity Testing of Single-Source Digraphs". In: *SIAM Journal on Computing* 27.1 (1998), pp. 132–169. DOI: 10.1137/S0097539794279626. eprint: https://doi.org/10.1137/S0097539794279626. URL: https://doi.org/10.1137/S0097539794279626.

[2] P. Bertolazzi et al. "Upward drawings of triconnected digraphs". In: *Algorithmica* 12.6 (Dec. 1994), pp. 476–497. ISSN: 1432-0541. DOI: 10.1007/BF01188716. URL: https://doi.org/10.1007/BF01188716.

[3] Mark De Berg and Amirali Khosravi. "Optimal Binary Space Partitions For Segments In The Plane". In: *International Journal of Computational Geometry & Applications* 22.03 (2012), pp. 187–205. DOI: 10.1142/S0218195912500045. eprint: http://www.worldscientific.com/doi/pdf/10.1142/S0218195912500045. URL: http://www.worldscientific.com/doi/abs/10.1142/S0218195912500045.

[4] Giuseppe Di Battista, ed. *Graph drawing : algorithms for the visualization of graphs*. An Alan R. Apt Book. Upper Saddle River, NJ: Prentice Hall, 1999. ISBN: 0-13-301615-3.

[5] Giuseppe Di Battista and Fabrizio Frati. "Drawing trees, outerplanar graphs, series-parallel graphs, and planar graphs in a small area". In: *Thirty Essays on Geometric Graph Theory*. Springer, 2013, pp. 121–165.

[6] M. R. Garey and D. S. Johnson. "Two-Processor Scheduling with Start-Times and Deadlines". In: *SIAM Journal on Computing* 6.3 (1977), pp. 416–426. DOI: 10.1137/0206029. eprint: https://doi.org/10.1137/0206029. URL: https://doi.org/10.1137/0206029.

[7] Michael R. Garey and David S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness*. New York: Freeman, 1979.

[8] Ashim Garg and Roberto Tamassia. "On the computational complexity of upward and rectilinear planarity testing". In: *Graph Drawing: DIMACS International Workshop, GD '94 Princeton, New Jersey, USA, October 10–12, 1994 Proceedings*. Ed. by Roberto Tamassia and Ioannis G. Tollis. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 286–297. ISBN: 978-3-540-49155-2. DOI: 10.1007/3-540-58950-3_384. URL: http://dx.doi.org/10.1007/3-540-58950-3_384.

[9] Ashim Garg and Roberto Tamassia. "Upward planarity testing". In: *Order* 12.2 (June 1995), pp. 109–133. ISSN: 1572-9273. DOI: 10.1007/BF01108622. URL: https://doi.org/10.1007/BF01108622.

[10] Michael D. Hutton and Anna Lubiw. "Upward Planar Drawing of Single-Source Acyclic Digraphs". MA thesis. University of Waterloo, 1990.

[11] Michael Jünger and Sebastian Leipert. “Level Planar Embedding in Linear Time”. In: *Graph Drawing: 7th International Symposium, GD'99 Štiřín Castle, Czech Republic September 15–19, 1999 Proceedings*. Ed. by Jan Kratochvíyl. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 72–81. ISBN: 978-3-540-46648-2. DOI: 10.1007/3-540-46648-7_7. URL: http://dx.doi.org/10.1007/3-540-46648-7_7.

[12] J. K. Lenstra and A. H. G. Rinnooy Kan. “Complexity of Scheduling under Precedence Constraints”. In: *Operations Research* 26.1 (1978), pp. 22–35. DOI: 10.1287/opre.26.1.22. eprint: http://dx.doi.org/10.1287/opre.26.1.22. URL: http://dx.doi.org/10.1287/opre.26.1.22.

[13] Achilleas Papakostas. “Upward planarity testing of outerplanar dags (extended abstract)”. In: *Graph Drawing: DIMACS International Workshop, GD '94 Princeton, New Jersey, USA, October 10–12, 1994 Proceedings*. Ed. by Roberto Tamassia and Ioannis G. Tollis. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 298–306. ISBN: 978-3-540-49155-2. DOI: 10.1007/3-540-58950-3_385. URL: https://doi.org/10.1007/3-540-58950-3_385.

[14] J.D. Ullman. “NP-complete scheduling problems”. In: *Journal of Computer and System Sciences* 10.3 (1975), pp. 384–393. ISSN: 0022-0000. DOI: http://dx.doi.org/10.1016/S0022-0000(75)80008-0. URL: http://www.sciencedirect.com/science/article/pii/S0022000075800080.

Appendix

A. Existence of Convex Cycle Embeddings

We will use this section to provide an idea how a proof of Conjecture 3.22 could look like. This will be mostly illustrated with pictures and will not be a formal proof. Figure A.1 shows three cycles with different non-convex embeddings. Being non-convex means that there is a vertical line intersecting the interior of the embedded graph at least twice, so there is always at least two *branches* of the cycle lying on top of each other. The figure shows the three possible ways how two branches can relate to each other: In Figure A.1a the branches do not overlap, both of them could be arbitrarily stretched to the right. In Figure A.1b the upper branch overlaps the lower one, so that the lower one cannot be arbitrarily stretched to the right. The reverse situation is shown in Figure A.1c.

Definition A.1 Let $G = (V, E)$ be a cycle and Γ be an upward planar embedding of G. A *segment* is a maximal path $P = (v_1, \ldots, v_k)$ of vertices, such that v_i lies left of v_{i+1} for all $1 \leq i < k$.

In case Γ is convex there are only two segments: One is formed by the lower sources and sinks, the other by the upper sources and sinks. In our example cycles in Figure A.1 we get four segments each, labeled A, B, C and D. We can transform these upward planar embeddings into the embeddings shown in Figure A.2. In Figure A.1a the "inner" segments B and C can just be flipped out, because the two branches of the embedding do not overlap and if segment C could be embedded above B between A and D, then C can also be

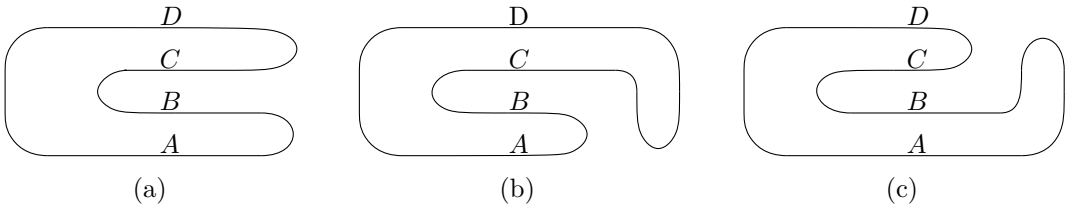

Figure A.1.: The three different ways two branches of a cycle can overlap (all of them can be mirrored horizontally). The four segments of each embedding are labeled with A, B, C and D. (a) The lower and the upper branch do not overlap. (b) The upper branch overlaps the lower one. (c) The lower branch overlaps the upper one.

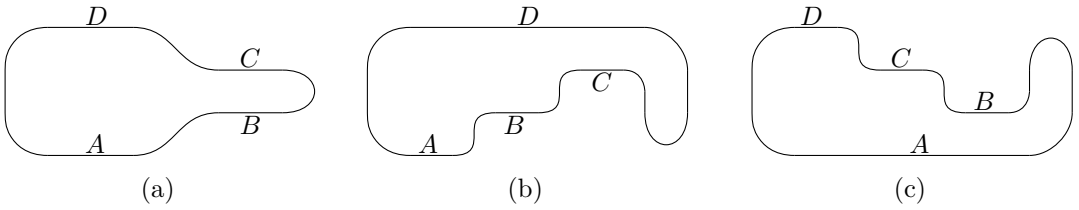

Figure A.2.: The shapes that the overlapping embeddings from Figure A.1 can be transformed into. The labeling of the segments is the same as in Figure A.1. (a) The embedding we get from Figure A.1a. (b) The embedding we get from Figure A.1b. (c) The embedding we get from Figure A.1c.

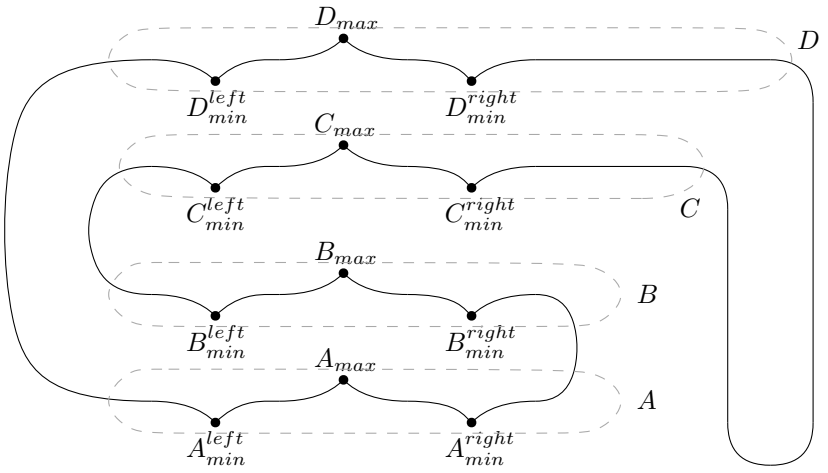

Figure A.3.: A cycle with a non convex embedding where the upper branch overlaps the lower one. Extremal vertices on each branch are labeled.

above B in the new embedding. To see why the transformations in Figure A.2b and A.2c work, we will consider the first case in more detail now. The other then follows from symmetry.

The embedding from Figure A.1b is shown again with added detail in Figure A.3. The shape is the same but we labeled some special vertices: A_{max} is the vertex with the highest y-coordinate in segment A in the given embedding, A_{min}^{left} and A_{min}^{right} are the vertices in segment A with the lowest y-coordinates in the embedding left and right of A_{max} respectively. Without loss of generality we assume that these are all unique. For the segments B, C and D the extremal vertices have been labeled similarly. The described transformation can now be done by flipping between segments A and B as well as between segments B and C while horizontally stretching segment D. This merges segments A, B and C into a single segment ABC, so that the resulting embedding is indeed convex. The result of this operation is shown in Figure A.4.

To prove that every cycle that has a non convex interval planar embedding also has a convex one we would now need to prove two statements:

1. The embedding we get from such a transformation is indeed interval planar if the original embedding was.

2. There is always four segments in any non convex embedding that allow one of the transformations described in Figures A.1 and A.2. We will leave this without a proof but will show a nontrivial example how this segments can be found.

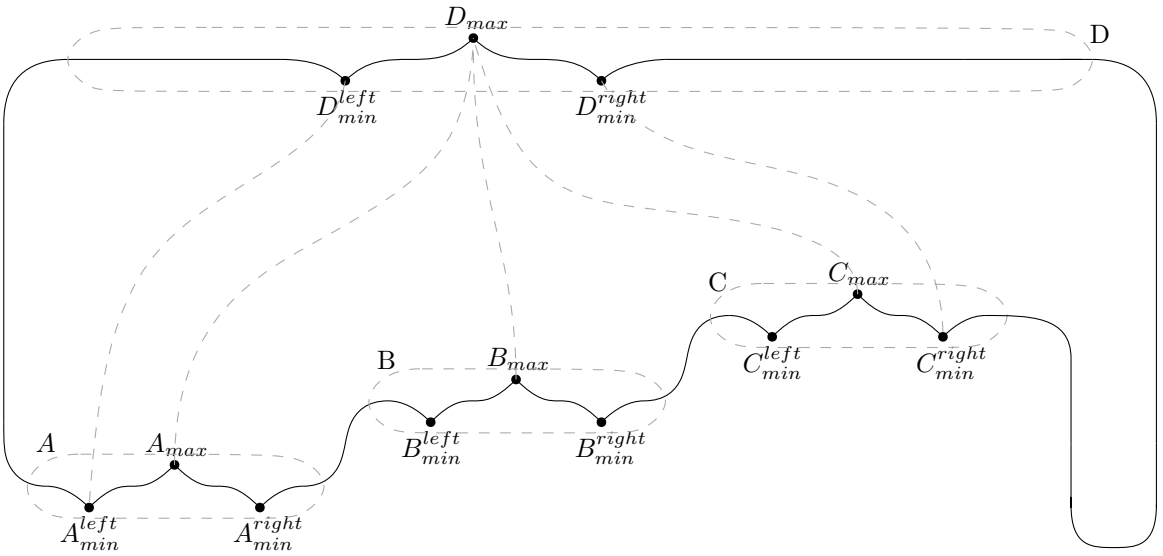

Figure A.4.: The convex expansion of the cycle given in Figure A.3. The dashed dummy edges show how the lower sinks and upper sources can be canceled.

If both statements are true, we can iteratively apply the described transformations, each time reducing the number of segments by two. Thus we would get to an embedding with just two segments, which is then convex.

Proof of Statement 1
Let $G = (V, E)$ be a graph with an interval planar embedding Γ as in Figure A.3. We know that A_{max} could be embedded below B_{max} in Γ. Similarly B_{max} could be embedded below C_{max} which could be embedded below D_{max} in Γ. By transitivity we then know that A_{max}, B_{max} and C_{max} can all be embedded below D_{max}, which is indicated by dashed dummy edges between them in Figure A.4. Since A_{max}, B_{max} and C_{max} were the highest vertices on their segments, every other vertex right of A_{max} and left of C_{max} can also be embedded below D_{max}.

What is left to show is that the vertices of segment D can still be embedded above the new segment ABC. From Γ we know that D_{min}^{left} could be above A_{min}^{left} and it still can in the new embedding. The same is true for D_{min}^{right} and A_{min}^{right}. Since these are the lowest vertices on D on either side of D_{max}, we can embed all other vertices on D above segment ABC as well. ■

We finish this section with an example. Given is the embedded cycle shown in Figure A.5a. We will apply above transformations until we get a convex embedding. The ten segments in this example are labeled $A, \ldots, J$.

1. The first step is not really a transformation, it is just some stretching along the x-coordinate to remove the deepest level of nesting branches. The result is shown in Figure A.5b. The resulting embedding still has the same ten segments.
2. Now consider segments A, B, E and F. They form two branches and the lower one overlaps the upper one on the left, so we are in the situation show in Figure A.1c (mirrored horizontally). Applying the transformation brings us to the embedding shown in Figure A.5c and merges segments B, E and F into a single segment BEF, so this embedding has only eight segments.
3. Segments J, G, D, C now form two non overlapping branches as in the situation from Figure A.1a. We can therefore merge J and G into a single segment JG and

merge C and D into a single segment CD. The resulting embedding is shown in Figure A.5d and has only six segments.

4. For the next transformation segments JG, I, H and CD are in the situation of two branches where the upper one overlaps the lower one as shown in Figure A.1c (mirrored horizontally). We merge I, H and CD into a single segment $IHCD$ and get the embedding shown in Figure A.5e, which now only has four segments.

5. Only one more transformation needs to be done. Segments JG, $IHCD$, BEF and A are as in Figure A.1c, forming two branches, where the lower one overlaps the upper one. We merge JG, $IHCD$ and BEF into a single segment $JGDCHIBEF$ and get the embedding in Figure A.5f. It now has only two segments and is indeed convex.

All of above transformations preserved interval planarity as we showed earlier. Therefore the embedding in Figure A.5f is interval planar if the one in Figure A.5a was.

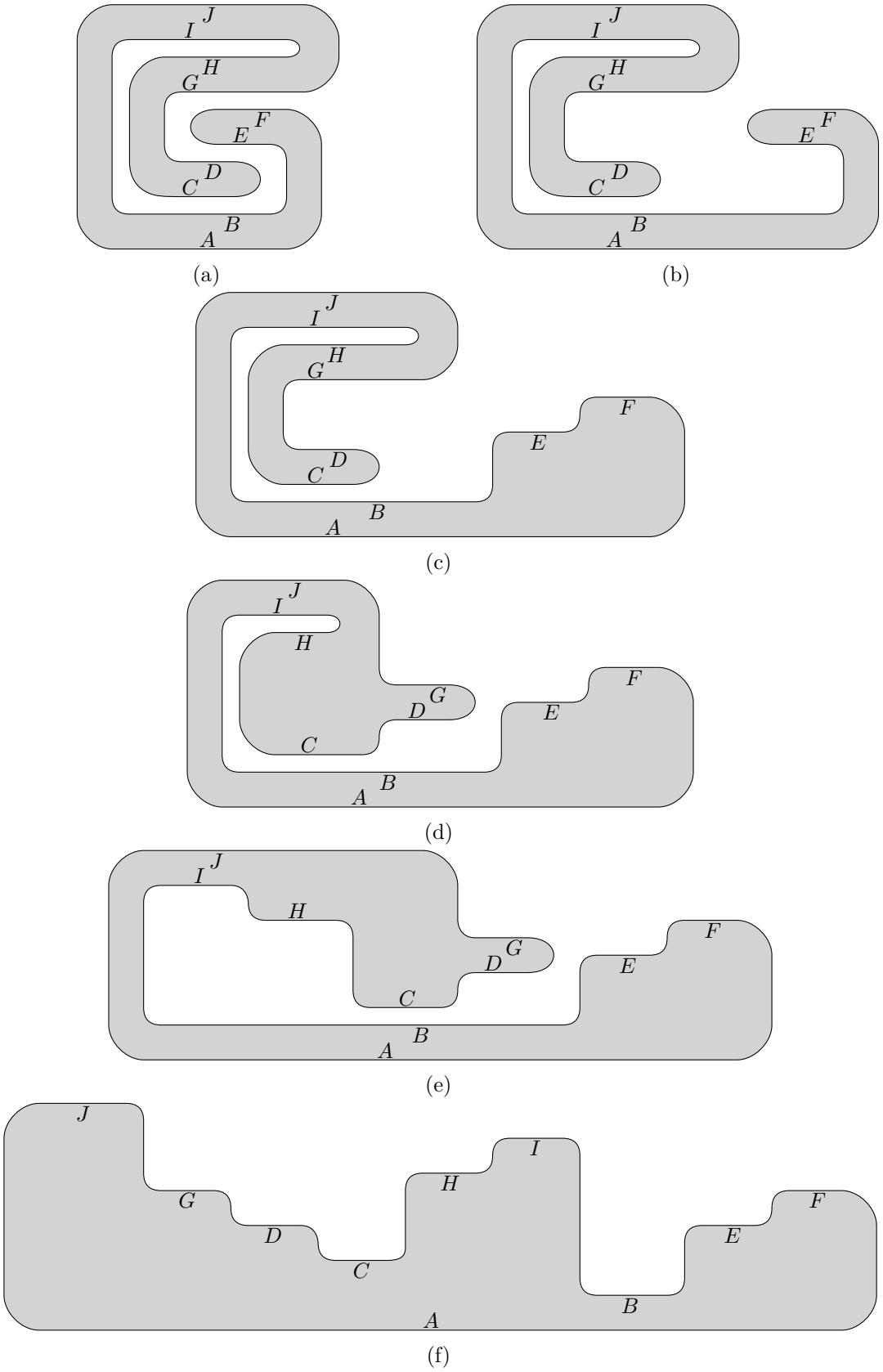

Figure A.5.: (a) A cycle with a non convex embedding and nested branches. (b) Stretching the lower branch reduces the nesting. (c) Joining segments B, E and F into a single segment BEF. (d) Joining segments C and D into a segment CD and joining segments J and G into a segment JG. (e) Joining segments I, H and CD into a segment $IHCD$. (f) Joining segments JG, $IHCD$ and BEF into a single segment $JGDCHIBEF$. The resulting embedding has only two segments, it is convex.