Karlsruhe Institute of Technology

Aligned Drawing of Plane Level Graphs

Master Thesis of

Lars Gottesbüren

At the Department of Informatics
Institute of Theoretical Informatics

Reviewers:	Prof. Dr. Dorothea Wagner
	Prof. Dr. Peter Sanders
Advisors:	Marcel Radermacher, M.Sc.
	Guido Brückner, M.Sc.

Time Period: 15th June 2017 – 14th December 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, 12th December 2017

Abstract

A level graph is a directed graph $G = (V, E)$ with an assignment $\mathrm{lvl} : V \to \{1, \dots, k\}$ of vertices to levels. In a level-planar drawing every vertex v is drawn on its level, the horizontal line $y = \mathrm{lvl}(v)$ and every edge is drawn as a y-monotone curve so that no two curves intersect in the interior. A pseudoline through a level-planar drawing is a simple y-monotone curve which intersects every edge at most once or fully covers it. It intersects every level exactly once, starts below the lowest level and ends above the topmost level. Let G be a graph with *topological* level-planar drawing and let $\mathcal{A}$ be an arrangement of pseudolines intersecting G. In this thesis we study (straight-line) *aligned level drawings* (Γ, A), where Γ is a level-planar (straight-line) polyline drawing of G, A is an arrangement of y-monotone straight lines homeomorphic to $\mathcal{A}$, and Γ and A intersect in the same way as G and $\mathcal{A}$. Mchedlidze et al. [MRR17] consider aligned drawings in planar graphs, and prove that every stretchable pseudoline arrangement intersecting a topologically embedded planar graph, where every edge is either fully covered or intersected at most once by a pseudoline, admit a straight-line aligned drawing.

Deciding whether $(G, \mathcal{A})$ and given A, admit a straight-line aligned level drawing (Γ, A), is the Straight-Line Aligned Level Drawing problem, for which we propose a polynomial-time algorithm, based on linear programming. Additionally, we present an asymptotically optimal algorithm for graphs with two levels, which solves the problem by reduction to a novel geometric problem. Furthermore, we show that there always is an aligned level drawing such that every edge is bent only where it is intersected by a pseudoline.

Moreover, we study a problem called Intersection Sequence Embedding. Its input is a level graph G without an embedding, a pseudoline arrangement $\mathcal{A}$ intersecting the levels of G, and a description of how $\mathcal{A}$ and G are supposed to intersect. The problem then asks whether G has a topological level drawing without edge crossings, which realizes the desired intersection. To solve it, we propose a linear-time reduction to testing level planarity, for which linear-time algorithms are known [JLM98].

Deutsche Zusammenfassung

Ein Level-Graph ist ein gerichteter Graph $G = (V, E)$ mit einer Funktion $\mathrm{lvl} : V \to \{1, \dots, k\}$, die jedem Knoten ein Level zuweist. In einer level-planaren Zeichnung wird jeder Knoten v auf der horizontalen Geraden $y = \mathrm{lvl}(v)$, seinem Level, gezeichnet. Außerdem wird jede Kante als y-monotone Kurve zwischen ihren Endpunkten gezeichnet, sodass sich keine der Kurven in ihrem Inneren kreuzen. Eine Pseudogerade durch eine level-planare Zeichnung ist eine einfache y-monotone Kurve, sodass jede Kante höchstens einmal gekreuzt wird oder die Kante ganz auf der Pseudogeraden liegt. Sie schneidet jedes Level genau einmal und durchquert die Zeichnung von unten nach oben. Sei G ein Graph mit einer gegebenen *topologischen* level-planaren Zeichnung und sei $\mathcal{A}$ eine Anordnung von Pseudogeraden, die G schneiden. In dieser Arbeit betrachten wir (geradlinige) *ausgerichtete Level-Zeichnungen* (aligned level drawings) (Γ, A), wobei Γ eine level-planare Zeichnung (geradlinig beziehungsweise Kantenzug) von G ist; und A ist eine Anordnung von Geraden, homöomorph zu $\mathcal{A}$, sodass sich Γ und A genauso schneiden wie G und $\mathcal{A}$. Mchedlidze et al. [MRR17] betrachten ausgerichtete Zeichnungen in allgemeinen planaren Graphen. Sie zeigen dass es zu jeder stretchbaren Anordnung von Pseudogeraden, die einen topologisch

eingebetteten planaren Graphen schneidet, eine geradlinige, ausgerichtete Zeichnung existiert, falls jede Kante entweder auf einer Pseudogeraden liegt oder von maximal einer Pseudogeraden geschnitten wird.

Straight-Line Aligned Level Drawing ist das Entscheidungsproblem, das, gegeben $(G, \mathcal{A})$ und A fragt, ob es eine geradlinige, ausgerichtete Level-Zeichnung (Γ, A) gibt. Basierend auf linearer Programmierung wird ein Polynomialzeit-Algorithmus für dieses Problem entwickelt. Außerdem wird ein asymptotisch optimaler Algorithmus, für den Fall dass der Level-Graph zwei Level hat, präsentiert. Zudem wird gezeigt, dass es immer eine ausgerichtete Zeichnung mit Knicken gibt, sodass jede Kante nur dort geknickt wird, wo sie von einer Pseudogeraden geschnitten wird.

Schlussendlich betrachten wir ein Problem namens Intersection Sequence Embedding. Es nimmt als Eingabe einen Level-Graph G ohne Einbettung, eine Anordnung an Pseudogeraden $\mathcal{A}$, die die Levels von G aber noch nicht G selbst schneidet. Außerdem Teil der Eingabe, ist eine Beschreibung wie $\mathcal{A}$ und G sich schneiden sollen. Das Problem stellt die Frage, ob G eine passende Einbettung hat, sodass die gewünschten Schnitteigenschaften zwischen $\mathcal{A}$ und G realisiert werden. Hierzu wird eine Linearzeit-Reduktion zum Testen von Level-Planarität gezeigt, wofür es wiederum Linearzeit-Algorithmen gibt, siehe [JLM98].

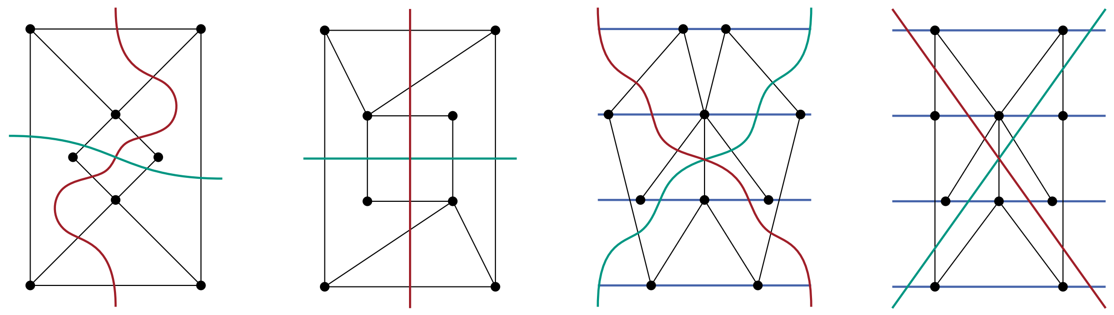
Contents

1. Introduction

Graph visualization is a widely used tool in data analysis, network analysis, VLSI design and software design. According to a study of Purchase [Pur97], an important property for visual comprehension of graph drawings is planarity or a small number of crossings for non-planar graphs. A common drawing type for data with an inherent hierarchical structure (such as branching timelines or management structure of a company) are level drawings. A level graph is a directed graph in which every vertex v is assigned to an integer y-coordinate $\mathrm{lvl}(v)$ such that $\mathrm{lvl}(u) < \mathrm{lvl}(v)$ for every directed edge (u, v). In a level-planar drawing every vertex v is drawn as a point with y-coordinate $\mathrm{lvl}(v)$ and edges are drawn as y-monotone curves between their vertices, such that no two curves cross in their interior. Level drawings with small crossing numbers have received a lot of attention, in particular due to the Sugiyama framework [STT81].

Two other properties of graph drawings that are desirable for visual comprehension are *aligning* a set $S \subset V$ of vertices by drawing them on a straight line or *separating* the disjoint parts $B \dot\cup C = V$ of a bipartition of the graph by a straight line. These were considered by Da Lozzo et al. [DLDF$^+$16] and Biedl et al. [BKM98], respectively. In the context of interactive graph drawing, a user may draw a curve through an existing graph drawing and let the computer decide whether the curve can be stretched into a straight line while maintaining planarity of the drawing. An important concept for this are pseudolines.

Let $G = (V, E)$ be a plane graph with a topological embedding. A pseudoline with respect to G is a simple curve such that every edge of G is intersected at most once or is fully contained in the pseudoline. Additionally, the pseudoline spans from and to infinity, i.e. starts and ends in the outer face of G. It turns out that a planar drawing of G in which the vertices of S are collinear exists, if and only if there is a pseudoline drawn through G which passes through all vertices of S. Similarly, G has a planar drawing in which B and C are separated by a straight line, if and only if there is a pseudoline through G which separates B and C. Mchedlidze et al. [MRR17] unify the aforementioned two properties and generalize the setting to more than one pseudoline. They introduce the notion of an *aligned graph* $(G, \mathcal{A})$ which comprises a plane embedded graph G and a *stretchable* pseudoline arrangement $\mathcal{A}$ with respect to G; see Figure 1.1(a) for an example. A *stretchable* pseudoline arrangement is homeomorphic to an arrangement of straight lines. Their subject of study are *aligned drawings* (Γ, A) where Γ is a planar polyline drawing of G and A is an arrangement of straight lines homeomorphic to $\mathcal{A}$ such that A and Γ intersect in the way as G and $\mathcal{A}$. In particular, they consider *straight-line aligned drawings*, where Γ is



(a) An aligned graph $(G, \mathcal{A})$.

(b) A straight-line aligned drawing (Γ, A) of $(G, \mathcal{A})$ in (a).

(c) An aligned level graph $(G, \mathcal{A})$.

(d) A straight-line aligned level drawing (Γ, A) of $(G, \mathcal{A})$ in (c).

Figure 1.1: Example of an aligned (level) graph $(G, \mathcal{A})$ and according straight-line aligned (level) drawing (Γ, A). Pseudolines $\mathcal{A}$ are drawn in red and green. The levels of the level graph G are drawn in blue. The graph G is in black.

a straight-line drawing of G. Figure 1.1(b) shows a straight-line aligned drawing of the aligned graph in Figure 1.1(a).

Our contribution is the consideration of straight-line aligned drawings in the realm of level-planar graphs. An *aligned level graph* is a tuple $(G, \mathcal{A})$ consisting of a level graph G topologically embedded in the plane and a topological arrangement of y-monotone pseudolines $\mathcal{A}$ intersecting G and its levels (from bottom to top). The analogous *aligned level drawing* of $(G, \mathcal{A})$ is the tuple (Γ, A), consisting of a planar polyline drawing Γ of G, and the straight line arrangement A such that $A + \text{lvl}(V)$, the arrangement combining A and the levels of V, is homeomorphic to $\mathcal{A} + \text{lvl}(V)$. As before, Γ and A intersect in the same way as G and $\mathcal{A}$.

Related Work

In the following we mention further related work. Biedl and Pennarun [BP16] study *non-aligned* drawings of planar graphs which aim to achieve the exact opposite of what we consider in this thesis, namely a planar drawing in the grid such that no two vertices lie in the same row or column. They show such a drawing exists in the $n \times n$-grid with at most $\frac{2n-5}{3}$ bends and such straight-line drawings exist in the $n^2 \times n^2$-grid.

Dujmović [Duj15] shows that every n-vertex planar graph has a planar drawing with $\Omega(\sqrt{n})$ collinear vertices, i.e. there is a stretchable pseudoline through $\Omega(\sqrt{n})$ vertices. In addition to the characterization of drawings in which S is drawn collinear, Da Lozzo et al. show that in planar graphs of treewidth 3 and triconnected cubic planar graphs there are drawings with $\Theta(n)$ collinear vertices, which is asymptotically optimal. For planar graphs of treewidth k they obtain a lower bound of $\Omega(k^2)$ on the number of possible collinear vertices.

Mchedlidze et al. [MRR17] study the Pseudoline Existence problem, which extends the work of Da Lozzo et al. [DLDF+16]. It asks, given a set $S \subset V$ of vertices to be drawn collinearly, is there a pseudoline with respect to G that collects all vertices of S. They show that it is $\mathcal{NP}$-complete by reduction from Hamiltonian Circuit and fixed-parameter tractable by reduction to K-Cycle, which is fixed-parameter tractable by a result of Wahlström [Wah13]. Additionally, they give a new proof of the characterization result of [DLDF$^+$16] and strengthen the result by showing that a fixed convex drawing of the outer face can be prescribed. Furthermore, they prove that every aligned graph $(G, \mathcal{A})$, in which every edge either fully lies on a pseudoline or is intersected by at most one pseudoline, has a straight-line aligned drawing. Deciding whether a pseudoline arrangement is stretchable is $\mathcal{NP}$-hard and even $\exists\mathbb{R}$-complete [Mnë88, Sho91]. Therefore Mchedlidze et al. [MRR17] assume a straight line arrangement A homeomorphic to $\mathcal{A}$ is given.

Additional related work is in the areas of level planarity and constrained drawings. Jünger et al.[JLM98] present a linear time algorithm to decide whether a given level graph is level-planar and Jünger and Leipert[JL99] extend this result to computing a combinatorial embedding, if one exists, in the same running time. Since their algorithm is complicated, Randerath et al. [RSB$^+$01] give a quadratic-time decision algorithm, Harrigan and Healy [HN08] present a quadratic-time decision and embedding algorithm and Fulek et al. [FPSŠ13] propose a quadratic-time decision algorithm based on a Hanani-Tutte characterization. Brückner and Rutter [BR17] consider Partial Level Planarity (PLP) and Constrained Level Planarity (CLP). In CLP for every level a partial ordering of the vertices on that level are given. The question is whether this partial ordering can be extended to a level-planar embedding. PLP considers the extension of a partial embedding to a complete level-planar embedding without modifying the partial embedding. Observe that PLP is a special case of CLP. The authors give a $O(n + kl)$ algorithm for CLP in single-source graphs, where n is the number of vertices, k is the number of levels and l is the size of the constraints. Additionally, they show that PLP is in general $\mathcal{NP}$-complete. Klemz and Rote [KR17] study Ordered Level Planarity, a variant of level planarity with a prescribed total order of the vertices on every level. The problem Ordered Level Planarity then asks whether a given level graph has a level-planar drawing with the prescribed order. They prove that Ordered Level Planarity is $\mathcal{NP}$-complete in general. In proper level graphs ($\mathrm{lvl}(v) - \mathrm{lvl}(u) = 1$ for every edge (u, v)) it can be solved in linear time, by testing whether the completely fixed drawing is crossing-free. The difficulty in non-proper level graphs lies in deciding where *long* edges ($\mathrm{lvl}(v) - \mathrm{lvl}(u) > 1$) cross levels.

For the sake of completeness, we point out that in Chapter 3 we consider results on constrained drawing in general planar graphs of Patrignani [Pat06] and Angelini et al.[ADF$^+$10], which we discuss then.

Contribution and Outline

In this thesis we study the problem of deciding whether a given aligned level graph $(G, \mathcal{A})$ and given straight line arrangement A has a straight-line aligned level drawing (Γ, A). We call this problem Straight-Line Aligned Level Drawing. Figure 1.1(d) shows a straight-line aligned level drawing of the aligned level graph in Figure 1.1(c). For the same reason as Mchedlidze et al. [MRR17], we assume A as part of the input. Chapter 2 formally introduces the basic concepts and notation. In Chapter 3 we consider drawing constraints in level graphs using linear programming. We introduce linear constraints to enforce straight-line drawings of edges and the embedding of a plane level graph. Furthermore we propose constraints to fix partial drawings of embedded graphs (to be extended to a drawing of the whole graph), fix turn directions between edges and force a face to be drawn as a convex polygon. The constraints to enforce the embedding and straight-line drawings from Chapter 3 are used in Chapter 4 to construct a linear program which computes straight-line aligned level drawings in polynomial time. Due to the nature of linear programming this can be combined with the above mentioned drawing constraints, into one framework, without additional effort. Note that, as opposed to [MRR17], our result works for edges which are intersected by more than one pseudoline. Additionally, we present a more efficient algorithm for graphs with two levels. The foundations for this are laid in Chapter 4 and in Chapter 5 we use them to devise the algorithm. In this context we consider a novel geometric problem, the MONOTONIC POLYGON HITTING SET problem, to which we reduce STRAIGHT-LINE ALIGNED LEVEL DRAWING with two-level graphs. We present two miscellaneous combinatorial results in Chapter 6. One is a characterization of pseudoline existence, similar to Mchedlidze et al. [MRR17]. In contrast to their result it is open whether PSEUDOLINE EXISTENCE is fixed-parameter tractable in level graphs. The other result concerns the INTERSECTION SEQUENCE EMBEDDING problem. Given a level graph $G = (V, E, \mathrm{lvl})$ without combinatorial embedding and a description of how a pseudoline arrangement shall intersect G, INTERSECTION SEQUENCE EMBEDDING is the problem of deciding whether there is a combinatorial embedding of G that realizes the desired intersections. We propose a linear-time reduction to testing level planarity.

2. Preliminaries

In this chapter we introduce a variety of concepts used throughout this thesis. Section 2.4 formally defines the Straight-Line Aligned Level Drawing problem, the main subject of this thesis.

2.1 Polygons

Define $\overline{st}$ as the straight line segment that joins two points s and t in the plane, and define $\overrightarrow{st}$ as the straight line that joins s and t. A *polygon* is a finite chain of straight line segments (polygonal chain) that forms a closed circuit. The straight line segments of the polygonal chain, are called its *edges* and the endpoints where two edges meet, are called *vertices.* A polygon is *simple* if it is not self-intersecting. An *open* polygon is a polygonal chain that is not a closed circuit. When we consider graphs and polygons simultaneously, we refer to the vertices of a polygon as polygon-vertices, if the text would otherwise be unclear. Similarly, we refer to edges of a polygon as polygon-edges. A simple polygon P partitions the plane into two regions, its interior and exterior, according to the Jordan curve theorem [Jor93]. P bounds the interior from the exterior. Regarding notation, we call the interior region interior(P) and the exterior region exterior(P). Note that the interior and the exterior are open sets, in the topological sense. P is referred to as the boundary of interior(P). Sometimes the term polygon is also used for the interior of a polygon and the polygon itself is referred to as boundary.

Definition 2.1 (Visibility)**.** *Two points p, q in a simple polygon P are visible from another, if and only if $\overline{pq} \cap \text{exterior}(P) = \emptyset$.*

A simple polygon is *convex* if every *internal angle* is less than or equal 180 degrees. Convexity is equivalently characterized by the property "every pair of two points in the interior or on the boundary are visible from another". Let P be a convex polygon and let v_l be the leftmost vertex of P. If there is more than one, choose the one with lowest y-coordinate. Similarly, choose v_r as the rightmost vertex of P. The *lower hull* P is the chain of all edges below the line connecting v_l and v_r. This definition excludes vertical edges from the lower hull on purpose. The *upper hull* is defined similarly as the chain of all edges above $\overrightarrow{v_l v_r}$. The *left hull* and *right* hull are defined similarly with the highest and lowest vertices. The *left-lower hull* of P is defined as the intersection of its left hull and lower hull. Similarly, we define the other hull combinations *left-upper hull, right-lower hull, right-upper hull.* By $|P|$ we denote the the number of vertices of P, which is equal to the number of its edges.

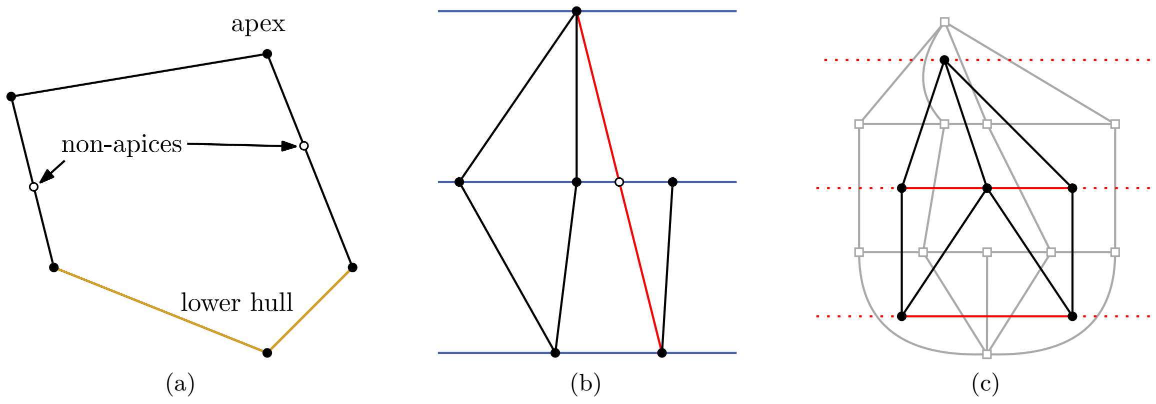


Figure 2.1: (a) shows the lower hull (orange) of a convex polygon. Additionally, it shows the distinction of apices and non-apex vertices.
In (b) the black edges form a proper level graph, the red edge is a long edge and the white disk is a level-crossing.
Finally (c) shows the infinite track-plane extension of a plane proper level graph. The level graph is in black, the horizontal red inter-level edges are due to the track-plane extension and the dotted horizontal lines stretching to infinity are due to the infinite track-plane extension. In gray we see the dual graph of the infinite track-plane extension.

Definition 2.2. *An apex (plural apices) of a convex polygon P is a vertex of P such that the inner angle between its two incident edges is strictly less than* 180 *degrees.*

Refer to Figure 2.1(a) for a visual distinction of apices and non-apex vertices. In the literature, two edges with internal angle equal to 180 degrees are usually seen as one edge. However, for the straight-line drawing algorithm for plane level graphs by Eades, Feng, Lin and Nagamochi [EFLN06], coming up in the next section, it is necessary to allow 180 degree angles at some vertices. We refrain from explaining the reasons behind this and simply use their result, see Theorem 2.5. In the other parts of this thesis we do not distinguish between apices and non-apex vertices.

2.2 Level Graphs

A level graph $G = (V, E, \mathrm{lvl})$ is a directed graph (V, E) with a mapping $lvl : V \to \{1, \dots, k\}$ where each vertex v is assigned to a y-coordinate $\mathrm{lvl}(v)$ such that for each edge $(u, v) \in E$ we have $\mathrm{lvl}(u) < \mathrm{lvl}(v)$. Although the definition is restricted to integer levels, for convenience we may use non-integer levels in constructions. We also write uv instead of (u, v). Sometimes we need edges that are not oriented or for which we do not want to fix whether $\mathrm{lvl}(u) < \mathrm{lvl}(v)$ or $\mathrm{lvl}(u) > \mathrm{lvl}(v)$, which we denote as $\{u, v\}$. An edge (u, v) is an *incoming* edge at v and an *outgoing* edge at u. By $N(v)$ we denote the set of neighbors of v, by $N^+(v)$ the neighbors through outgoing edges and by $N^-(v)$ the neighbors through incoming edges. A vertex with no incoming edge is called *source*, a vertex with no outgoing edge is called *sink*. The horizontal line $y = \mathrm{lvl}(v)$ is called the *level* of vertex v. The set $\mathrm{lvl}(V)$ is referred to as the levels in G and $k := |\,\mathrm{lvl}(V)|$ is referred to as the number of levels. The *span* of an edge $e = uv \in E$ is defined as $\mathrm{span}(e) := \mathrm{lvl}(v) - \mathrm{lvl}(u)$ (recall $\mathrm{lvl}(v) > \mathrm{lvl}(u)$). Edges of span greater than one are called *long*. Level graphs with no long edges are called *proper*.

A level graph is called *level-planar* if it has a planar drawing in which every vertex v is drawn at its assigned y-coordinate $\mathrm{lvl}(v)$ and every edge is drawn as a y-monotone curve. A *combinatorial embedding* $\prec$ of a proper level graph is an ordering $\prec_i$ of the vertices on each level i, sometimes denoted $\prec = \{\prec_i \mid i = 1, \dots, k\}$. We refer to $\prec_i$ as the *level-order* on level i. In order to define combinatorial embeddings for level graphs with long edges we need to introduce an object type that indicates where a long edge $uv \in E$ crosses the intermediate levels $\mathrm{lvl}(u) + 1, \dots, \mathrm{lvl}(v) - 1$.

Definition 2.3. *Subdivide a long edge $uv \in E$ with vertices on the intermediate levels $\mathrm{lvl}(u) + 1, \dots, \mathrm{lvl}(v) - 1$ so that it forms a path of proper edges. The* level-crossing *of e on level i is the subdivision vertex on level i. Its purpose is to indicate where in the total order of vertices and other level-crossings on level i, the edge e is supposed to cross level i.*

Figure 2.1(b) shows a level-crossing and the distinction between proper and level graphs with long edges. Similar to the proper case, a *combinatorial embedding* $\prec$ of a level graph with long edges is an order of the vertices and level-crossings on each level. A level graph $G = (V, E, \mathrm{lvl})$ with fixed combinatorial embedding $\prec$ is called a *plane level graph*. We also denote it by $G = (V, E, \mathrm{lvl}, \prec)$. Observe that a combinatorial embedding of a level graph fixes its outer face, as opposed to general planar graphs. A drawing of a level graph G is *straight-line* if each edge is drawn as a straight line segment.

An *st-plane level graph* is a plane level graph with one source s and one sink t. Observe that $\mathrm{lvl}(s)$ must be the lowest level occupied by a vertex and $\mathrm{lvl}(t)$ must be the highest level occupied by a vertex. A *triangulated* level graph is a plane level graph where each face except for the outer face is triangular.

Straight-Line Drawing of Plane Level Graphs

We briefly touch on a result due to Eades et al. [EFLN06] on straight-line drawings of plane level graphs with long edges. In this thesis it is used in Section 4.3.

Definition 2.4. *Let H be a triangulated st-plane level graph H, and let P be a straight-line drawing of its outer facial cycle C. P is called feasible for H, if P is convex and for every chord uz of C, both of the two paths joining u and z on C have a vertex drawn as an apex of P.*

Theorem 2.5 (Eades, Feng, Lin, Nagamochi [EFLN06])**.** *Let H be a triangulated st-plane level graph and let polygon P be a straight-line drawing of its outer facial cycle. If P is feasible for H there exists a planar straight-line drawing of H with outer face P.*

Track Planarity

The following definitions concern the notion of track planarity, which we use in Section 4.2 and Section 6.2. A track graph is a generalization of a level graph, where the condition $\mathrm{lvl}(u) < \mathrm{lvl}(v)$ is relaxed to $\mathrm{lvl}(u) \leq \mathrm{lvl}(v)$ for the directed edge (u, v). An edge $e = (u, v)$ is called *inter-level* if $\mathrm{lvl}(u) < \mathrm{lvl}(v)$ and *intra-level* if $\mathrm{lvl}(u) = \mathrm{lvl}(v)$. A track graph is *track-planar* if it has a planar drawing such that the intra-level edges are drawn horizontally, on their respective levels, vertices are drawn on their respective levels and the inter-level edges are drawn as strictly y-monotone curves.

Definition 2.6. *Let $G = (V, E, \prec)$ be a plane proper level graph. Its* track-plane *extension is the track-plane graph obtained from G by adding an intra-level edge between any pair of consecutive vertices in $\prec_i$ for $i = 1, \dots, k$.*

Definition 2.7. *The* infinite track-plane extension *of a plane proper level graph G is its track-plane extension augmented by two additional inter-level edges on each level. One inter-level edge stretching to a vertex at positive x-infinity and one stretching to negative x-infinity is added on the right, respectively left end of that level.*

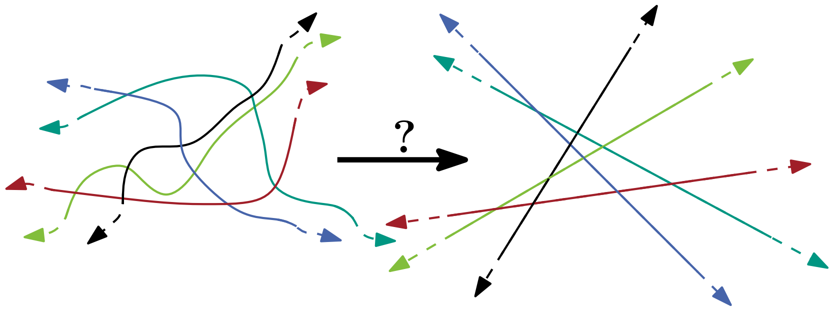


Figure 2.2: On the left: an arrangement of pseudolines, on the right: a stretched version of the left side. Figure provided by Marcel Radermacher [Rad15].

Some visual aid is in order, for which we refer to Figure 2.1(c). The edges stretching to x-infinity on both sides of each level allow speaking of the unbounded spaces between levels as unbounded faces. This in turn allows the definition of a certain type of dual graph we need, where instead of one dual vertex for the outer face, there are two for the unbounded spaces (left,right) in the space between two consecutive levels. Additionally, there is a dual vertex s below the bottom level and a dual vertex t above the top level, making the dual a st-plane track graph. Note that the infinite track-plane extension is not a graph per se but its dual graph is still well-defined.

2.3 Pseudolines and Stretchability

Definition 2.8 (Pseudoline)**.** *A pseudoline is a simple y-monotone curve in the Euclidean plane stretching from and to infinity.*

A *pseudoline arrangement* $\mathcal{A} = \{\mathcal{R}_1, \dots, \mathcal{R}_z\}$ is a set of pseudolines, pairwise meeting at most once and crossing where they meet. The points where pseudolines meet are called *pseudoline-intersections*. The segments of one pseudoline $\mathcal{R}$ between consecutive pseudoline-intersections along the trajectory of $\mathcal{R}$ are called the *pseudosegments* of $\mathcal{R}$. Pseudoline arrangements generalize line arrangements in the sense that the topological and combinatorial properties of line arrangements are preserved (intersection properties, partition into segments between intersections) whereas the straightness aspect is discarded ([TOG04]). Note that in the literature, pseudolines are usually defined as images of straight lines under homeomorphisms in the projective plane such that pseudolines in an arrangement pairwise intersect exactly once (be it an intersection at infinity). Throughout this thesis we do not require two pseudolines in an arrangement to intersect, unless explicitly stated.

One pseudoline partitions the Euclidean plane into two halfplanes and a pseudoline arrangement $\mathcal{A}$ partitions the Euclidean plane into a *cell complex*, where $\mathcal{A}$ bounds the cells. Denote by cells($\mathcal{A}$) the cell complex into which $\mathcal{A}$ partitions the plane. See the left side of Figure 2.2 for an arrangement of pseudolines and its cell complex.

Definition 2.9 (Stretchable Pseudoline Arrangement)**.** *A pseudoline arrangement $\mathcal{A}$ is* stretchable *if and only if some homeomorphism $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$ maps every $\mathcal{R} \in \mathcal{A}$ to a straight line. The arrangement $\{\varphi(\mathcal{R}) \mid \mathcal{R} \in \mathcal{A}\}$ is called a* stretching *of $\mathcal{A}$.*

Figure 2.2 shows a stretchable pseudoline arrangement and a stretching. The Stretchability problem asks, whether a given pseudoline arrangement in which pseudolines pairwise cross exactly once, is stretchable. The version of stretchability with pseudolines in the projective plane was proven to be $\exists\mathbb{R}$-complete by Mnëv [Mnë88] and $\mathcal{NP}$-hard by Shor [Sho91]. Herein $\exists\mathbb{R}$ is a complexity class, introduced by Schaefer [Sch09]. Deciding truth in *the existential theory of the reals* to $\exists\mathbb{R}$ is the equivalent of satisfiability to $\mathcal{NP}$. The existential theory of the reals is the set of true statements of the form

$$\exists x_1, x_2, \ldots, x_n : \varphi(x_1, x_2, \ldots, x_n)$$

where φ is a quantifier-free and negation-free Boolean formula over the signature $(0, 1, +, *, <)$ interpreted over the real numbers (Schaefer [Sch09]).
By a result of Canny [Can88], the complexity class $\exists\mathbb{R}$ is decidable in PSPACE. Furthermore $\mathcal{NP}$ is contained in $\exists\mathbb{R}$. Schaefer [Sch09] mentions that STRETCHABILITY is still $\exists\mathbb{R}$-complete for pseudolines defined as y-monotone curves in the Euclidean plane as in Definition 2.8.

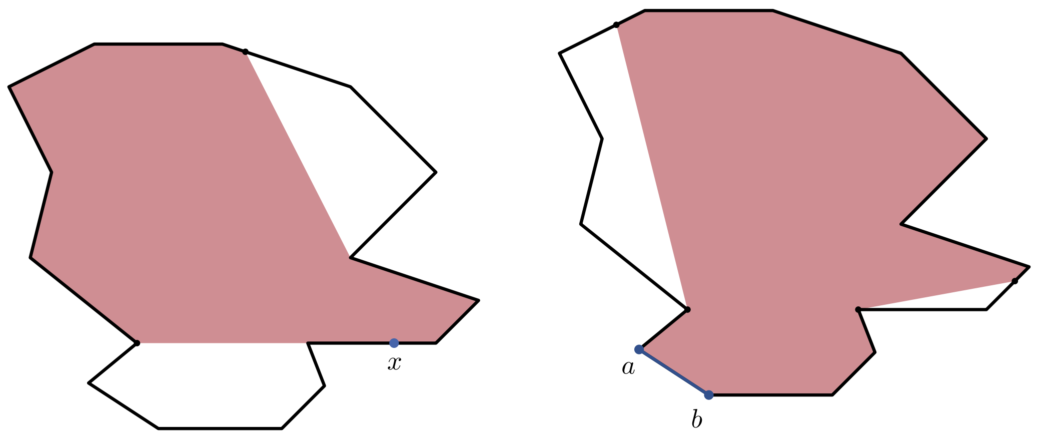
2.4 Aligned Level Drawing

Let $G = (V, E, \text{lvl}, \prec)$ be a plane level graph, topologically embedded in the Euclidean plane. A pseudoline $\mathcal{R}$ with respect to G is a simple y-monotone curve through the topological embedding of G, intersecting some of the vertices, levels and edges of G in a certain order. It spans from negative to positive infinity, intersects every level exactly once, in the order $1, \ldots, k$, and is inherently oriented upwards. As $\mathcal{R}$ is y-monotone, it intersects every level either in one vertex, in one level-crossing or in an interval between pairs of the former two. Any edge either lies fully on $\mathcal{R}$ or is intersected by $\mathcal{R}$ in at most one point (incident vertex or interior). A pseudoline arrangement $\mathcal{A}$ with respect to G is a pseudoline arrangement intersecting the topological embedding of G. A tuple $(G, \mathcal{A})$ of a plane level graph $G = (V, E, \text{lvl}, \prec)$ and a pseudoline arrangement with respect to G is called an *aligned level graph*. In addition to vertices, levels and edges, a pseudoline also intersects other pseudolines of the arrangement in a certain order. We call the arrangement combining the pseudolines of $\mathcal{A}$ with the levels $\text{lvl}(V)$ their *combined arrangement* $\mathcal{A}+\text{lvl}(V)$. If $\mathcal{A}+\text{lvl}(V)$ is stretchable, $A + \text{lvl}(V)$ is a stretching of $\mathcal{A} + \text{lvl}(V)$ and Γ is a polyline drawing of G under $\prec$, then (Γ, A) is called an *aligned level drawing* of $(G, \mathcal{A})$, if and only if Γ and A intersect in the same way as G and $\mathcal{A}$. More formally, let φ be the function that maps every vertex to its point in Γ, maps every edge to its polyline in Γ, maps every pseudoline of $\mathcal{A}$ to its corresponding straight line of A and maps every level to itself. Furthermore, for every pseudoline $\mathcal{R} \in \mathcal{A}$, let $\text{iseq}_G(\mathcal{R})$ be the sequence of pseudolines, vertices, levels and edges which $\mathcal{R}$ intersects in $(G, \mathcal{A})$. Similarly, let $\text{iseq}_\Gamma(\varphi(\mathcal{R}))$ be the sequence of lines of A, vertices, levels and edges which $\varphi(\mathcal{R})$ intersects in Γ. Then (Γ, A) is an aligned level drawing of $(G, \mathcal{A})$ if and only if for every pseudoline $\mathcal{R} \in \mathcal{A}$ we have $\varphi(\text{iseq}_G(\mathcal{R})) = \text{iseq}_\Gamma(\varphi(\mathcal{R}))$. If Γ is a straight-line drawing we call (Γ, A) a *straight-line aligned level drawing.*

We used the notion of homeomorphisms in the definition of stretchable arrangements. A homeomorphism may exchange the left and right halfplane of all pseudolines at once, or may reverse the order of levels. This is not problematic as the suitably mirrored straight line arrangement (around the vertical axis, respectively horizontal axis) complies with the required orientation of levels and pseudolines.

Definition 2.10. *Given* $(G, \mathcal{A})$ *and* A*,* STRAIGHT-LINE ALIGNED LEVEL DRAWING *is the problem of deciding whether* $(G, \mathcal{A})$ *has a straight-line aligned level drawing* (Γ, A)*.*

We assume A as part of the input because STRETCHABILITY is $\mathcal{NP}$-hard, even $\exists\mathbb{R}$-complete. It is possible to formulate a similar decision problem for aligned level drawings, which are not forced to be straight-line. However, in Section 4.3 we will see that every aligned level graph has an aligned level drawing. The remaining question is how many bends are required. Since we do not give a bend-minimization algorithm we do not formulate this optimization problem.



(a) Point visibility polygon. (b) Weak edge visibility polygon.

Figure 2.3: Polygon P and exemplary Point Visibility Polygon $\mathrm{PV}(P, x)$ in (a) as well as exemplary Weak Edge Visibility Polygon $\mathrm{WEV}(P, (a, b))$ in (b).

An edge is called *i-crossed* if it is intersected by i pseudolines and does not lie on a pseudoline. It is called *aligned* if it lies fully on a pseudoline and it is called *free* if it is intersected by zero pseudolines. Similarly, a vertex on a pseudoline is called *aligned* and a vertex not on a pseudoline is called *free*. We denote the number of pseudolines intersecting an i-crossed edge e by $\zeta(e) := i$. If e is free, we have $\zeta(e) = 0$. For an aligned edge e we set $\zeta(e) = 0$, even if there are other pseudolines intersecting e. In Chapter 4 and Chapter 5 we use $\zeta(e)$ to denote the number of bends on edge e in an aligned level drawing, as well as in the running time of algorithms for Straight-Line Aligned Level Drawing. Aligned edges have a unique drawing in aligned level drawings and it is bend-free. Due to their unique drawing they furthermore do not contribute to the complexity of Straight-Line Aligned Level Drawing and thus cost no additional running time. These are the reasons for setting $\zeta(e) = 0$ for aligned edges.

Level Segments

The intersections of pseudolines in $\mathcal{A}$ and a level, partition the level into segments. If there is no intersection on the level, then it is partitioned into one segment. If there is at least one intersection then the level has two unbounded segments (leftmost and rightmost) and the remaining segments are bounded by two intersections. In an aligned level drawing, a vertex must be drawn in such an interval if it is free. If it is aligned, it must be drawn on the intersection of its level and the pseudoline that aligns the vertex. For vertex u, we denote by level-segment(u) the interval or intersection of $\mathrm{lvl}(u)$ in $A + \mathrm{lvl}(V)$ where u must be placed in an aligned level drawing. We call level-segment(u) the level-segment of u. Similarly for a level-crossing d, we denote by level-segment(d) its level-segment.

2.5 Visibility in Polygons

Recall Definition 2.1, which defines visibility between two points in a simple polygon P. The *point visibility polygon* $\mathrm{PV}(P, x)$ consists of those parts of P that are visible from a point x. Intuitively, we can imagine $\mathrm{PV}(P, x)$ as the part of P that is illuminated by direct light, when placing a point light source at x; ignoring refraction. There are a variety of algorithms that solve this problem efficiently in $O(|P|)$ time [EA81, Lee83, JS87]. In Figure 2.3(a) the point visibility polygon of some point x in some polygon P is visualized.

Weak edge visibility extends the notion of point visibility in the intuitive sense that the light source, formerly a point, is now a light-emitting line segment of P.

Definition 2.11 (Weak Edge Visibility)**.** *Let $s = (a, b)$ be an edge of P. The* weak edge visibility polygon $\text{WEV}(P, (a, b))$ *is the maximal subpolygon of P, for which each point of* $\text{WEV}(P, (a, b))$ *can see at least one point in (a, b).*

Figure 2.3(b) shows the weak edge visibility polygon of some segment $s = (a, b)$. There are algorithms that compute $\text{WEV}(P, (a, b))$ efficiently in $O(|P| \log |P|)$ time [LL86, EG85, O'R87] and even in $O(|P| \log \log |P|)$ time by combining the $O(|P| \log \log |P|)$ triangulation algorithm from Tarjan and Van Wyk [TVW88] and the linear-time weak edge visibility algorithm in triangulated polygons by Guibas et al [GHL+87].

We discuss a $O(|P| \log |P|)$ horizontal-line sweep algorithm from the book *Art Gallery Theorems and Algorithms* by O'Rourke [O'R87]. In Section 4.4.2 we modfify the algorithm and use its notion of *critical lines*, which is the reason we describe it in such detail. The description very closely follows that in the book.

First, P is rotated so that s lies horizontally with the interior of P above s. Then the vertices are sorted increasingly, by y-coordinate. Edges are oriented upwards and for simplicity we assume that no edge other than s lies horizontally. This allows the distinction whether the interior of P lies to the left or right of an edge.

Horizontal Sweepline Datastructures

The main idea is to sweep P with a horizontal sweep-line H that moves upwards through P, starting at s. At any stage H intersects a set of edges $e_1, e_2, \ldots, e_z$ from left to right, such that the edges with odd index bound P from the left and the edges with even index bound P from the right. We call those with odd index *left* edges and those with even index *right* edges. A consecutive pair e_i, e_{i+1} with e_i being a left edge, bounds a *window* on H, such that the interval on H between the edges lies in the interior of P. Such a window is called *visibility window*, if some part of the interval is visible from s. Each window has a separate left and right boundary.

The edges intersected by H and the visibility windows are stored using a balanced search tree for $O(\log |P|)$ time access, insertions and deletions. Only the order in which edges intersect H are stored, not the actual intersections. An intersection can be reconstructed in constant time, when needed. Updating the intersections in every advancement of H would yield quadratic running time.

Visibility Windows Datastructures

Each point x on H that is visible from s sees an interval of points on s. Denote by $L(x)$ the line segment connecting x to its leftmost visible point on s and by $R(x)$ the line segment connecting to the rightmost visible point. Additionally, let $c[L(x)]$,$c[R(x)]$ be the point of contact between the boundary of P and $L(x)$,$R(x)$ respectively. If there are multiple points, choose the one closest to x on the right boundary of W for $R(x)$, respectively on the left boundary for $L(x)$ of W. For example, in Figure 2.4, the critical line R_0 has a point of contact on the left boundary of W but $c[R_0]$ is taken on the right boundary.

Each visibility window is partitioned into intervals where $c[L(x)]$ does not change, in the interval. The points $x_l, \ldots, x_0$ where $c[L(x)]$ does change, determine the *left critical lines* $L_l, \ldots, L_0$ with $L_i = L(x_i)$, crossing H in this left-to-right order. Similarly, the points where $c[R(x)]$ changes determine the *right critical lines* $R_0, \ldots, R_r$, crossing H in this left-to-right order. See Figure 2.4 for an illustration.

For each visibility window, the critical lines and their according points of contact are stored in two separate balanced search trees L and R. This allows the construction of $L(x)$

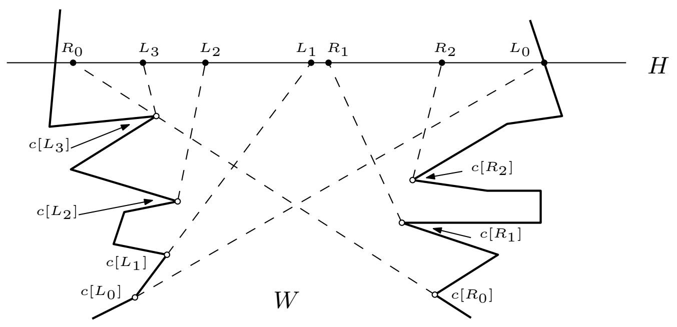


Figure 2.4: Critical lines and their points of contact in a visibility window W.

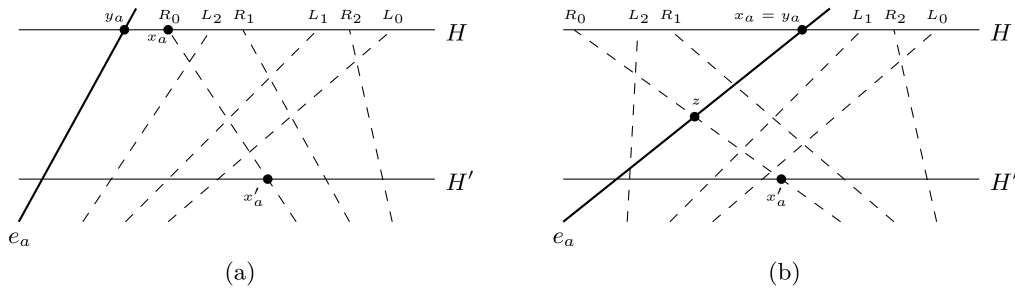


Figure 2.5: Output and window updates due to advancing H.

and $R(x)$ in $O(\log |P|)$. If x lies between L_i and L_{i+1} on H, then $c[L(x)] = c[L(x_i)]$ and we can construct $L(x)$ from that. As above, only the order in which critical lines intersect H are stored in the search tree. Again, updating every intersection of critical lines with H in every advancement of H would yield quadratic running time. Reconstructing one intersection is in constant time.

Advancing the Sweepline

In the following we describe the updates to the search trees and the output of visible segments, that occur in one advancement of the sweepline. Let H be the sweepline at the next vertex x. Locate x as a vertex of an edge e in the edges intersected by H. Recall that not every window is a visibility window. If x does not lie in the interior or on the boundary of a visibility window, immediately move the sweepline on to the next vertex.

Otherwise x lies in some window W and three actions are taken, as described in the following three paragraphs. Note that we describe them only for the left window boundaries and the right window boundaries are treated analogously. Let e_a be the left bounding edge of W and let H' be the sweepline state, the last time that W was updated. First the visible segments in W between H and H' are output, then updates on W due to advancing H are performed and finally updates to the edges intersected by H and window updates due to x are performed.

Output of Visible Segments

Let y_a be the intersection of e_a and H and let x_a be the leftmost visible point on H in W. If y_a lies left of R_0 then x_a is the intersection of H and R_0, otherwise $x_a = y_a$. Additionally,

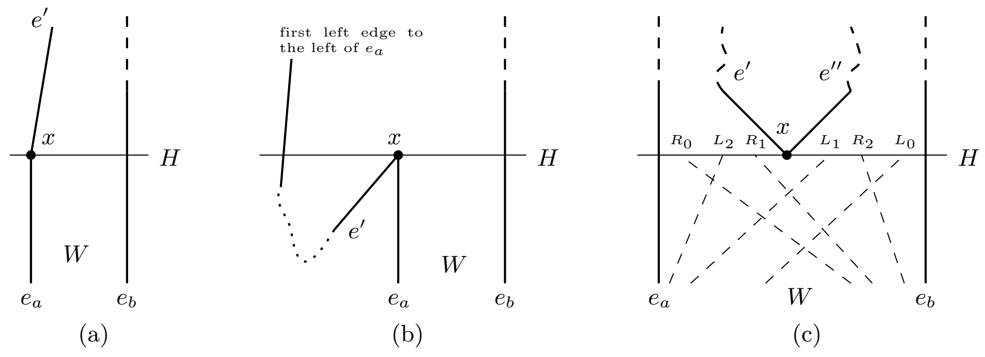


Figure 2.6: Window updates due to x.

let x'_a be the leftmost visible point of H' in W. Due to the next step (Window Updates due to Advancing H), we will know that x'_a is the intersection of H' and R_0, as R_0 was constructed thusly.

Distinguish the two cases $x_a = y_a$ and y_a lies to the left of x_a. If y_a lies to the left of x_a, then no segment of e_a between H and H' is visible, as x_a lies on R_0. Therefore output the segment $x'_a x_a$, see Figure 2.5(a). If on the other hand $y_a = x_a$, then y_a lies right of R_0 and thus e_a must intersect R_0 somewhere between H and H', in some point z. Then the two segments $x'_a z$ and $z x_a$ are output; see Figure 2.5(b).

Window Updates due to Advancing H

If x_a is right of L_0 or $R_0 = L_0$ the window W is closed. If $x_a \neq y_a$ no updates to L and R are made, see Figure 2.5(a). If $x_a = y_a$ and x_a lies between R_i and R_{i+1} on H, the lines $R_0, \dots, R_i$ are deleted from R and a new R_0 that connects x_a and $c[R_i]$ is inserted. Similarly, if x_a does not lie left of L_l and thus lies between L_{i+1} and L_i, the critical lines $L_l, \dots, L_{j+1}$ are deleted from L and a new L_{j+1} connecting x_a and $c[L_i]$ (not $c[L_{i+1}]$) is inserted into L. See Figure 2.5(b).

Window Updates due to x

The last part of the advancement step consists of updates to the edges intersected by H and the resulting partition in windows. We distinguish two cases.

Case 1. x is the upper endpoint of e_a, i.e. the left boundary.

Then W is bounded by a different edge in the next step. Let e' be the other edge incident to x. If e' has the interior of P to its right, then e' bounds W in the next step, see Figure 2.6(a). Otherwise the first edge to the left of e_a on H, that has interior(P) to its right is chosen as the next boundary of W, see Figure 2.6(b).

Case 2. x is the upper endpoint of the right boundary. Analogous to Case 1.

Case 3. x is a lower endpoint of both its incident edges e' and e''.

In this case W is split into two windows W' and W''. Let e_b be the edge bounding W from the right. Then W' is bounded by e_a from the left and by e' from the right. W'' is bounded by e'' from the left and e_b from the right. The critical lines L, R are split between W' and W''. Let x be located between R_i and R_{i+1} and between L_j and L_{j+1}. Then W' gets $R_0, \dots, R_i$ and $L_l, \dots, L_{j+1}$, whereas W'' gets $R_{i+1}, \dots R_r$ and $L_j, \dots, L_0$. Finally, $L(x)$ and $R(x)$ are added to W' and W''. See Figure 2.6(c).

Initialization

After describing the sweepline advancement steps, we show how to initialize the sweepline and the search trees. We start with H on the height of $s = (a, b)$. Recall that a and b are the leftmost, respectively rightmost points on s. Let e_a, e_b be the edges closest to a, b respectively, out of the edges intersected by H which are incident to one vertex above H. It is possible that a is a lower endpoint of e_a but this is not necessarily the case. There is one window W, bounded by e_a and e_b. Then let x_a be the intersection of e_a and H, similarly x_b. L is initialized with two critical lines $L_0 = ax_b, L_1 = ax_b$, similarly R with $R_0 = bx_a, R_1 = bx_b$. See Figure 2.7 for an illustration.

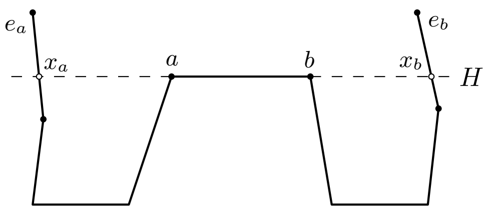


Figure 2.7: Initialization of sweepline, windows and critical lines.

Note that we omit a detailed description for parts visible below the edge. A point-visibility query from each endpoint of the edge suffices. However, in our application for weak edge visibility (Chapter 4) we have no parts below the edge.

2.6 Linear Programming

Linear programming is a subfield of optimization, where a linear objective function is optimized under linear inequality constraints. More formally let $\mathbf{x} \in \mathbb{R}^n$ be a vector of variables, whose value is to be set by the linear program solver, let $\mathbf{c} \in \mathbb{R}^n$ be a vector of real-valued cost function coefficients, let $A \in \mathbb{R}^{m \times n}$ be a matrix of real-valued coefficients and let $b \in \mathbb{R}^m$ be a vector. Then a linear program can be formalized as:

Maximize $\mathbf{c}^{\mathrm{T}}\mathbf{x}$ such that $A\mathbf{x} \leq \mathbf{b}$.

Therein $\mathbf{c}^{\mathrm{T}}\mathbf{x}$ is the objective function and $A\mathbf{x} \leq \mathbf{b}$ are the m linear inequality constraints. An objective function can be minimized by maximizing $-\mathbf{c}$.

Karmarkar's algorithm [Kar84] solves a linear program in $O(m^{3.5}L^2)$ time, where L is the number of bits in the input and m is the number of constraints.

Theorem 2.12. *Linear programs can be solved in $O(m^{3.5}L^2)$ time.*

In this thesis, we use linear programs without objective functions. We call these *linear constraint programs* and write them as $\mathcal{LCP} = (\mathit{Var}, \mathit{Con})$ where Var is a set of variables and Con is a set of linear inequalities over Var. The linear inequalities are of the form $\sum_{Z \in \mathit{Var}} a_Z \cdot Z \leq c$ with $a_Z, c \in \mathbb{R}$.

3. Linear Constraint Programming and Constrained Drawing Problems

In Chapter 4 we will use linear constraint programming to devise an algorithm for Straight-Line Aligned Level Drawing. In Section 3.1 of this chapter we lay the foundations for that. Since aligned level drawings are a form of constrained drawings, we also consider some constrained drawing problems in this chapter. Combining the discussed constrained drawing problems with Straight-Line Aligned Level Drawing is immediate, due to the nature of linear constraint programming. In Section 3.2 we discuss partial drawing extension problems and in Section 3.3 we discuss constraints on the slopes of straight-line edge drawings. In particular we show how to constrain a face of a plane level graph to be drawn as a convex polygon. Since linear constraint programming is inherently simultaneous in the sense that any sets of linear constraints can be combined, this approach is especially suited for interactive drawing purposes where multiple constraints on a drawing shall be combined and a combined algorithm is difficult to envision.

3.1 Straight-Line Plane Level Graph Drawing

We start with a straight-forward linear constraint program for straight-line drawings of a plane level graph $G = (V, E, \mathrm{lvl}, \prec)$. A straight-line drawing of G is represented by a set of variables X_u, one for the x-coordinate of every vertex $u \in V$. The y-coordinate $y_u = \mathrm{lvl}(u)$ is fixed by its level. Recall that the embedding is given as a set of level-orders $\prec = \{\prec_1, \dots, \prec_k\}$ with $\prec_i = \langle d_1, \dots d_{n_i}\rangle$ denoting the order of vertices and level-crossings on level i. Introduce for each level-crossing d a variable X_d for its x-coordinate. The set of constraints to draw G with embedding $\prec$ is defined by the following equation:

$$\mathrm{Embedding}(\prec) = \bigcup_{i=1}^{k} \bigcup_{j=1}^{n_i-1} \{X_{d_j} < X_{d_{j+1}}\}\,. \tag{3.1}$$

In a straight-line drawing the level-crossings and endpoints of an edge are collinear.

Definition 3.1 (x-slope)**.** *The x-slope of a line $\overrightarrow{st} = \{s + \lambda(t - s) \mid \lambda \in \mathbb{R}\}$ is defined as*

$$M_{\overrightarrow{st}} = \frac{1}{y_t - y_s} \cdot (x_t - x_s)\,.$$

Usually the slope of the line $\{s + \lambda(t-s) \mid \lambda \in \mathbb{R}\}$ is defined as $\frac{y_t - y_s}{x_t - x_s}$. This definition is unsuitable for our purpose, as the slope is not linear in x_t and x_s. Furthermore, representing a vertical line would require $\frac{y_t - y_s}{x_t - x_s} = \infty$.

We force collinearity of level-crossings and endpoints of an edge $e \in E$, by requiring every line segment joining two level-crossings or endpoints on consecutive levels to have the same x-slope M_e.

Definition 3.2. *The x-slope M_e of the drawing of $e = (s,t) \in E$ is defined as*

$$M_e = \frac{1}{y_t - y_s} \cdot (X_t - X_s)\,,$$

which is a linear combination of X_t and X_s.

Let $d_{e,1}, \ldots, d_{e,r(e)}$ be the level-crossings of edge $e = (s,t) \in E$, where $r(e) = \operatorname{lvl}(t) - \operatorname{lvl}(s) - 1$. The following equation constitutes the constraints for a straight-line drawing of e.

$$\begin{aligned} \text{StraightEdge}(e) = &\left\{ \frac{X_{d_{e,1}} - X_s}{y_{d_{e,1}} - y_s} = M_e \right\} \\ &\cup \left(\bigcup_{j=1}^{r(e)-1} \left\{ \frac{X_{d_{e,j+1}} - X_{d_{e,j}}}{y_{d_{e,j+1}} - y_{d_{e,j}}} = M_e \right\} \right) \\ &\cup \left\{ \frac{X_t - X_{d_{e,r(e)}}}{y_t - y_{d_{e,r(e)}}} = M_e \right\} \end{aligned} \tag{3.2}$$

Thus the set of constraints for a straight-line drawing of G are expressed by

$$\text{StraightLineDrawing}(E) = \bigcup_{e \in E} \text{StraightEdge}(e)\,. \tag{3.3}$$

If we assume that levels are spaced evenly with distance one, as level graphs were defined in Section 2.2, the constraints are simplified to

$$\begin{aligned} \text{StraightEdge}(e) = &\left\{ X_{d_{e,1}} - X_s = M_e \right\} \\ &\cup \left(\bigcup_{j=1}^{r(e)-1} \left\{ X_{d_{e,j+1}} - X_{d_{e,j}} = M_e \right\} \right) \\ &\cup \left\{ X_t - X_{d_{e,r(e)}} = M_e \right\}\,. \end{aligned} \tag{3.4}$$

The linear constraint program $\mathcal{LCP}$-FE-SL incorporates both the embedding (FE for fixed embedding) and straight-line edge drawings (SL) for drawing level graphs $G = (V,E)$ with fixed embedding $\prec$.

$$\begin{aligned} \mathcal{LCP}\text{-FE-SL} &= (\mathit{Var}\text{-FE-SL}, \mathit{Con}\text{-FE-SL}) \\ &= (\{X_u \mid u \in V\}, \text{StraightLineDrawing}(E) \cup \text{Embedding}(\prec)) \end{aligned} \tag{3.5}$$

It is the foundation for the upcoming constrained drawing problems and a linear constraint program, described in Section 4.5, for Straight-Line Aligned Level Drawing. As the running time Kamarkar's algorithm contains a factor L^2, where L is the number of bits in the input, we analyze here how many bits are required to represent $\mathcal{LCP}$-FE-SL.

The "right side" of every linear constraint in $\mathcal{LCP}$-FE-SL is zero, i.e. the vector $\mathbf{b}$ from the matrix formulation $A\mathbf{x} \leq \mathbf{b}$ of linear programs is the zero-vector. Its representation requires a constant number of bits. Define l as the number of level-crossings. The number of variables is $l + |V|$, so identifying one variable requires $O(\log(l + |V|))$ bits. The representation of the constraints Embedding($\prec$) (Equation 3.1) requires $O((l + |E|)\log(l + |V|))$ bits as every such constraint uses two variables with coefficients from $\{1, -1\}$.

We assume levels are spaced evenly with distance one. The representation of the constraints StraightEdge(e) (Equation 3.4) requires $O((l + |E|)\log|V|\log k)$ bits. In every constraint from StraightEdge(e) the term M_e involves two variables, each with coefficient $\frac{1}{y_t - y_s}$ or $\frac{-1}{y_t - y_s}$. There are $2\binom{k}{2}$ such values, i.e. we require $O(\log k)$ bits to represent them. The coefficients of the other two variables are 1 or -1.

Lemma 3.3. *$\mathcal{LCP}$-FE-SL can be represented using $O((l + |E|)\log(l + |V|)\log k)$ bits.*

3.2 Partial Drawings

Partial drawings have been of particular interest to graph drawing researchers. These typically consist of a fixed geometric drawing or combinatorial embedding of a subgraph $H \subset G$ or subset $V' \subset V$ that shall be extended to a planar drawing of the supergraph G.

Definition 3.4 (Partially Embedded Planarity)**.** *Given a planar graph G, a subgraph H of G and a combinatorial embedding $\mathscr{H}$ of H,* Partially Embedded Planarity *is the problem of deciding if there is a combinatorial embedding $\mathscr{G}$ of G such that $\mathscr{G}$ restricted to the vertices and edges of H yields $\mathscr{H}$.*

Note that Partially Embedded Planarity does not constrain edges to be straight-line segments. A linear-time algorithm for Partially Embedded Planarity was proposed by [ADF+10].

Definition 3.5 (Partial Drawing Extensibility)**.** *Given a planar graph $G = (V, E)$, a subset $V' \subset V$ and fixed coordinates $\psi(v') \in \mathbb{R}^2$ for $v' \in V'$,* Partial Drawing Extensibility *is the problem of deciding if there are coordinates for the remaining vertices $V \setminus V'$ so that the resulting straight-line drawing is planar.*

It is $\mathcal{NP}$-complete as proven by [Pat06]. Note that in Partial Drawing Extensibility the combinatorial embedding of G is not fixed, only that of $G[V]$.

Definition 3.6 (Fixed Embedding Partial Drawing Extensibility)**.** *Given a planar graph $G = (V, E)$ with fixed combinatorial embedding, a subset $V' \subset V$, fixed coordinates $\psi(V')$,* Fixed Embedding Partial Drawing Extensibility *is the problem of deciding whether there are coordinates for the vertices $V \setminus V'$ so that the resulting straight-line drawing is planar and is a drawing of the given combinatorial embedding.*

The complexity of Fixed Embedding Partial Drawing Extensibility (FEPDE) in general planar graphs is unknown at this time. For level graphs, it is straight-forward to modify $\mathcal{LCP}$-FE-SL to solve Fixed Embedding Partial Drawing Extensibility. Let $\psi : V' \to \mathbb{R}$ be the mapping from vertices of V' to their fixed x-coordinates. It can be achieved by replacing all occurrences of the variable $X_{v'}$ by $\psi(v')$ for all $v' \in V'$. For the sake of consistency, we also note the according linear constraints, which could be added, instead of replacing the variables by constants.

$$\text{FEPDE}(V, V', \psi) = \{X_{v'} = \psi(v') \mid v' \in V'\} \tag{3.6}$$

3.3 Constraints on x-Slopes

In this section we discuss problems that admit linear constraints by constraining the x-slopes of edge drawings.

3.3.1 Turn Directions

Consider two incident edges $e_1 = (s,t) \in E$ and $e_2 = \{t,u\} \in E$ with $\mathrm{lvl}(s) < \mathrm{lvl}(t)$, whereas whether $\mathrm{lvl}(t) < \mathrm{lvl}(u)$ or $\mathrm{lvl}(t) > \mathrm{lvl}(u)$ is undetermined. Just for this section, orient e_1 from s to t and e_2 from t to u. The turn direction at the shared vertex t is *left* if the drawing of e_2 extends into the left halfplane of the drawing of e_1, and the turn direction is *right* if the drawing extends into the right half plane and straight if they are collinear.

The constraint for a right turn is constituted by the following equation.

$$\mathrm{RightTurn}(e_1 = (s,t), e_2 = \{t,u\}) = \begin{cases} M_{e_2} > M_{e_1} & \text{, if } \mathrm{lvl}(u) > \mathrm{lvl}(t) \\ M_{e_1} < M_{e_2} & \text{, if } \mathrm{lvl}(u) < \mathrm{lvl}(t) \end{cases} \tag{3.7}$$

The case distinction is due to the fact that in the definition of $M_{e_2} = \dfrac{X_u - X_t}{y_u - y_t}$ we assume $\mathrm{lvl}(t) < \mathrm{lvl}(u)$, i.e. orient the edge $\{t,u\}$ as (t,u).

For a left turn we only need to swap the conditions.

$$\mathrm{LeftTurn}(e_1 = (s,t), e_2 = (t,u)) = \begin{cases} M_{e_2} < M_{e_1} & \text{, if } \mathrm{lvl}(u) > \mathrm{lvl}(t) \\ M_{e_1} > M_{e_2} & \text{, if } \mathrm{lvl}(u) < \mathrm{lvl}(t) \end{cases} \tag{3.8}$$

The constraint for a straight turn is just enforcing collinearity.

$$\mathrm{StraightTurn}(e_1, e_2) = \{M_{e_2} = M_{e_1}\} \tag{3.9}$$

Note that if $\mathrm{lvl}(u) < \mathrm{lvl}(t)$, a straight turn would break the combinatorial embedding $\prec$.

3.3.2 Convex Faces

In this section we discuss the following problem. Given a cycle $C = (v_0, \ldots, v_{|C|-1})$ of G, is there a drawing of G so that C is drawn as a convex polygon. A polygon is convex, if and only if, when traversing its boundary in counter-clockwise order, it turns left at every polygon-vertex. Assume $(v_0, \ldots, v_{|C|-1})$ is the counter-clockwise order of the vertices around the cycle. This is well-defined in level graphs with fixed embedding. The constraints for a convex drawing of C are defined by the following equation.

$$\mathrm{ConvexCycle}(C) = \bigcup_{i=0}^{|C|-1} \mathrm{LeftTurn}(v_{i \bmod |C|}, v_{i+1 \bmod |C|}, v_{i+2 \bmod |C|}) \tag{3.10}$$

3.3.3 Fixed Angles Between Edges and Horizontal Lines

In this section we show how to prescribe the angle between a straight-line drawing Γe of $e = (s,t) \in E$ and the horizontal line through s. Let α be the angle to prescribe. From the unit circle we know that $X_t - X_s = \cos(\alpha)$ and $y_t - y_s = \sin(\alpha)$. Thus the constraint for fixing the angle between $\Gamma(e)$ and the horizontal line through s is given by the following equation

$$\mathrm{FixedAngle}(e = (s,t), \alpha) = \{M_e = \cot(\alpha)\}\,. \tag{3.11}$$

Even though $\cot(\alpha)$ is not linear in α, it is a precomputable constant.

4. Aligned Level Drawing

In this chapter we mostly discuss the Straight-Line Aligned Level Drawing problem. First we give an example of a plane proper level graph and pseudoline arrangement which does not admit a straight-line aligned level drawing, see Section 4.1. Then in Section 4.2 we show that for "small" pseudoline arrangements and any plane proper level graph, a straight-line aligned level drawing always exists. Here small means up to two pseudolines or parallel pseudolines. Section 4.3 shows that any aligned level graph has an aligned level drawing with $\zeta(e)$ bends on every edge e, assuming the necessary condition that the combined arrangement of levels and pseudolines is stretchable. Subsequently we consider straight-line aligned level drawings from the perspective of single edges in Section 4.4. In particular we introduce the notion of *intersection preservation* for single edges, which encapsulates the following property. A drawing (Γ, A) of $(G, \mathcal{A})$ is a straight-line aligned level drawing if and only if every edge is straight, drawn intersection-preserving and Γ is a drawing of the combinatorial embedding $\prec$ of G. This yields a polynomial time algorithm to decide whether $(G, \mathcal{A})$ has a straight-line aligned level drawing, see Theorem 4.17. It is based on linear constraint programming and builds upon results from the previous chapter. Additionally we describe an efficient representation for all straight-line intersection-preserving drawings of a single edge. This is used in Chapter 5 to build a faster algorithm (compared to the linear constraint program) to decide Straight-Line Aligned Level Drawing for graphs with two levels.

4.1 Pappus Configuration

In this section, we show an example of a proper aligned level graph without a straight-line aligned level drawing. It is based on the non-Pappus pseudoline arrangement (see Figure 4.1), which is a non-stretchable arrangement of nine pseudolines. Its name is due to Pappus of Alexandria as its non-stretchability is due to the non-Pappus arrangement violating Pappus' hexagon theorem, as observed by Levi [Lev26].

Theorem 4.1 (Pappus' Hexagon Theorem)**.** *Let A, B, C and a, b, c be two triples of collinear points. Additionally, let X be the intersection of Ab and Ba, let Y be the intersection of Ac and Ca and finally let Z be the intersection of Cb and Bc. Then X, Y, Z are collinear.*

If the pseudolines were stretched, the pseudoline connecting X and Z but avoiding Y violates Pappus' hexagon theorem and therefore this arrangement is not stretchable.

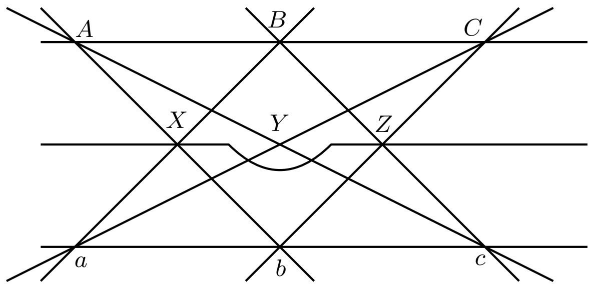


Figure 4.1: Non-Pappus pseudoline arrangement.

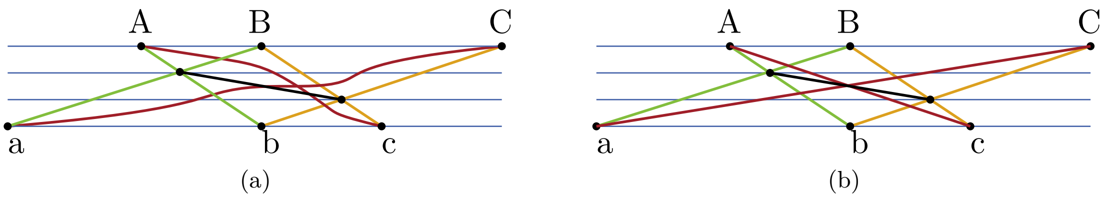


Figure 4.2: Pappus configuration. In the stretched version (right), the single edge (black) must run through the crossing of Ac and Ca (red) as opposed to the pseudoline version (left), where the pseudolines Ac and Ca are drawn as non-straight curves.

Based on the non-Pappus arrangement we construct a proper aligned level graph, called *Pappus configuration*, with four levels and six pseudolines which does not have a straight-line aligned level drawing. Figure 4.2(a) shows that graph and pseudoline arrangement. The pseudoline arrangement consists of the pseudolines Ab, Ac, Ba, Bc, Ca, Cb with

- A, B, C aligned by level 4
- a, b, c aligned by level 1
- Ab, Ba intersecting on level 3
- Bc, Cb intersecting on level 2
- and Ac, Ca intersecting between levels 2 and 3.

The graph consists of two vertices u, v and a single edge uv with v on the intersection of Ab, Ba and u on the intersection of Bc, Cb. The edge uv runs below the intersection of Ac, Ca. By Pappus' hexagon theorem the edge uv must run through Ac, Ca in a stretching of the combined arrangement of pseudolines and levels, see Figure 4.2(b). Observe that the combined arrangement of levels and pseudolines itself is stretchable. Another interesting observation is that in the Pappus configuration, two levels play the role of pseudolines, by aligning triples of points.

4.2 Small Pseudoline Arrangements and Proper Level Graphs

In this section we consider straight-line aligned level drawings in proper level graphs (i.e. edges with span one) and "small pseudoline arrangements". By small we mean either *parallel* pseudolines or arrangements of two pseudolines. We show that in these cases the pseudoline arrangements always admit a straight-line aligned level drawing due to the nature of proper level graphs, independent of the graph.

Definition 4.2. *A pseudoline arrangement $\mathcal{A}$ with respect to a plane level graph is parallel, if none of the pseudolines intersect.*

Lemma 4.3 (Parallel Pseudolines). *Let $G = (V, E, \mathrm{lvl}, \prec)$ be a plane proper level graph and let $\mathcal{A} = \{\mathcal{R}_1, \dots, \mathcal{R}_z\}$ be a parallel pseudoline arrangement with respect to G. Then $(G, \mathcal{A})$ has a straight-line aligned level drawing.*

Proof. Order the pseudolines $\mathcal{R}_1, \dots, \mathcal{R}_z$ from left to right (x-coordinate), i.e. let $\mathcal{R}_1$ be the left-most and $\mathcal{R}_z$ be the right-most. This is possible since they are parallel and we can use the smallest x-value of a curve as the sorting criterion. As $\mathcal{R}_1, \dots, \mathcal{R}_z$ do not overlap on vertices it is possible to draw each vertex aligned by $\mathcal{R}_i$ on the vertical line with x-coordinate i without conflict, i.e. draw each $\mathcal{R}_i$ vertically as the line $R_i = \{(x, y) \in \mathbb{R} \mid x = i\}$.

Place the free vertices between the vertical grid lines at equidistant x-coordinates with respect to $\prec$. In fact any positioning of the free vertices that fulfills $\prec$ suffices. Thus the resulting straight-line drawing is level-planar.

The order of vertices traversed by each R_i is preserved trivially due to y-monotonicity. Edges intersect pseudolines only between their endpoints' levels. The order in which edges intersect pseudolines is solely determined by $\prec$. Therefore the resulting straight-line drawing is isomorphic to $(G, \mathcal{A})$. $\square$

Corollary 4.4 (One Pseudoline). *Any one pseudoline with respect to a plane proper level graph admits a straight-line aligned level drawing.*

This corollary follows directly from Lemma 4.3. It is a version of Theorem 4.9 (see the next section) for plane proper level graphs. In particular, this corollary implies that given a set $S \subset V$ of vertices that are supposed to be collinear, this is possible if and only if there is a pseudoline through the vertices of S with respect to the plane proper level graph G.

The following theorem shows that any arrangement of two pseudolines with respect to a plane proper level graph always admits a straight-line aligned level drawing.

Theorem 4.5 (Two Pseudolines). *Let $G = (V, E, \mathrm{lvl}, \prec)$ be a plane proper level graph and let $\mathcal{A} = \{\mathcal{R}_1, \mathcal{R}_2\}$ be a pseudoline arrangement of two pseudolines, with respect to G.*

Proof. We distinguish three cases: whether the pseudolines intersect and if they do, whether the intersection lies on one level or between two consecutive levels.

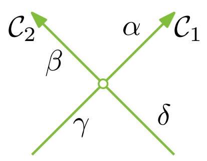
Case 1. $\mathcal{R}_1, \mathcal{R}_2$ are parallel. See Lemma 4.3.

For the other two cases, partition each pseudoline into two halves, split at their intersection. Denote the upper ray of $\mathcal{R}_1$ by α, the lower ray by γ, the upper ray of $\mathcal{R}_2$ by β and the lower ray by δ (see Figure 4.3(a)). The cell complex of $\{\mathcal{R}_1, \mathcal{R}_2\}$ consists of four cells, which we call *quadrants*. They are named North (bounded by β, α), West (β, γ), South (γ, δ) and East (δ, α) (see Figure 4.3(c)).

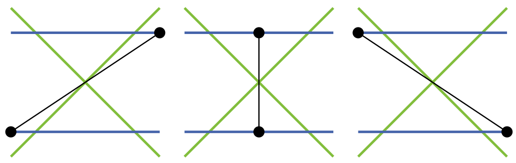
Case 2. $\mathcal{R}_1 \cap \mathcal{R}_2$ lies on a level, say level i.

Draw the stretching R_1 of $\mathcal{R}_1$ as a straight line with slope 1 and R_2, the stretching of $\mathcal{R}_2$ as a straight line with slope -1, such that their intersection lies on level i. Place every vertex v aligned by $\mathcal{R}_i$ at the intersection of R_i and $\mathrm{lvl}(v)$.

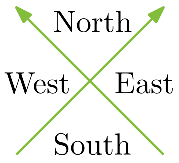
It is easy to see that the straight-line drawing, obtained by greedily or equidistantly placing every free vertex inside its quadrant and with respect to $\prec$, is an aligned drawing of $(G, \{\mathcal{R}_1, \mathcal{R}_2\})$. The order $\prec$ ensures that R_1, R_2 intersect edges in the same order as $\mathcal{R}_1, \mathcal{R}_2$. Since $\mathcal{R}_1 \cap \mathcal{R}_2$ lies on a level and G is proper, it is already clear from the levels whether an edge intersects $R_1/\mathcal{R}_1$ on α or γ and is independent of the pseudolines. Therefore the resulting drawing is topologically equivalent to $(G, \{\mathcal{R}_1, \mathcal{R}_2\})$.



(a) Naming convention of the rays of $\mathcal{R}_1, \mathcal{R}_2$.



(b) All configurations of $e, \mathcal{R}_1, \mathcal{R}_2$ in Case 3.1 of the proof of Theorem 4.5.



(c) Quadrants.

Figure 4.3

Case 3. $\mathcal{R}_1 \cap \mathcal{R}_2$ lies between two levels $i, i+1$.

Draw the stretching R_1 of $\mathcal{R}_1$ as a straight line with slope 1 and R_2, the stretching of $\mathcal{R}_2$ as a straight line with slope -1, such that their intersection is half-way between levels $i, i+1$. Place every vertex v aligned by $\mathcal{R}_i$ at the intersection of R_i and $\text{lvl}(v)$.

The vertices on levels below i and above $i+1$ can be placed as in Case 2, i.e. greedily with respect to $\prec$ and in their respective quadrants. This is done only after the vertices of levels $i, i+1$ are placed. Notice that edges between levels $(i-1, i)$ and $(i+1, i+2)$ are drawn topologically equivalent to $(G, \{\mathcal{R}_1, \mathcal{R}_2\})$, if and only if their endpoints are in their respective quadrants and $\prec$ is respected.

The vertices on levels $i, i+1$ require greater care, since, as opposed to Case 2, an arbitrary quadrant- and embedding-respecting positioning may let an edge intersect the wrong ray of a stretched pseudoline. Since we consider two consecutive levels, the edges we consider, possess a well-defined left-to-right ordering, induced by their endpoint ranks in $\prec_i, \prec_{i+1}$. Denote this ordering by $\langle e_1 \prec_E e_2 \prec_E \ldots \rangle$. Additionally denote the set of edges between levels $i, i+1$ by $E_i = \{e = uv \mid \text{lvl}(u) = i \wedge \text{lvl}(v) = i+1\}$.

Let G' be the infinite track-plane extension of G, recall Definition 2.7. G' partitions the space between levels $i, i+1$ into a sequence of quadrangular and triangular faces, ordered by adjacency. The pseudolines traverse this sequence, starting and ending somewhere, either in a strict left-to-right or strict right-to-left fashion. We distinguish two cases: whether the pseudoline intersection $\mathcal{R}_1 \cap \mathcal{R}_2$ lies on an edge or in a face.

Case 3.1. $\mathcal{R}_1 \cap \mathcal{R}_2$ lies on an edge $e = (u, v)$. Draw e with u, v in their respective quadrants. Since e meets both pseudolines at their intersection, e has to intersect both pseudolines.

If u is in the South quadrant then v is in the North quadrant. In this case any edge to the left of e can only intersect β and γ, any edge to the right only α and δ.

If u is in the West quadrant then v is in the East quadrant. In this case any edge to the left of e can only intersect α and β, any edge to the right only γ and δ.

If u is in the East quadrant then v is in the West quadrant. In this case any edge to the left of e can only intersect γ and δ, any edge to the right only α and β. See Figure 4.3(b) for all three configurations of $e, \mathcal{R}_1, \mathcal{R}_2$.

Let $e_2 \in \{e' \in E_i \mid e' \prec_E e \vee e' = e\}$ be an edge that is either e or to the left of e and let e_1 be the edge directly left of e_2, if it exists. By the planarity of the embedding, e_1 can only intersect the same rays as e_2 or fewer. If e_2 intersects a ray and e_1 does not, any edge to the left of e_1 cannot intersect that ray either. There are at most two such transitions. It suffices to draw the edges to the left of e that intersect the same rays "close together".

Drawing the edges to the right of e is analogous.

Case 3.2. $\mathcal{R}_1 \cap \mathcal{R}_2$ lies in a face f of G'.

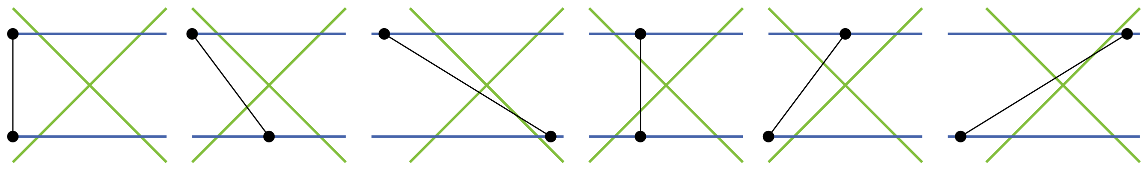


Figure 4.4: All configurations of $e_l, \mathcal{R}_1, \mathcal{R}_2$ in Case 3.2 of the proof of Theorem 4.5.

If it is unbounded on both sides, there is no edge between levels i and $i+1$ and we are done.

Then, without loss of generality assume the face is bounded by an edge $e_l = (u_l, v_l)$ to the left and an edge $e_r = (u_r, v_r)$ to the right. Figure 4.4 shows all six possible configurations of $e_l, \mathcal{R}_1, \mathcal{R}_2$. As in Case 3.1 all edges to the left of e_l can only intersect the same or fewer rays as e_l, as evidenced by Figure 4.4. Again there are at most two transitions and drawing edges that intersect the same rays close together again suffices. We first draw e_l and e_r. If they are incident, their shared vertex is placed first. On a "clean canvas" it is obviously possible to let the drawings of e_l and e_r intersect the same rays as e_l and e_r do. Then the edges to the left of e_l are drawn in decreasing order in $\prec_E$, in such a way that edges that intersect the same rays are grouped closely together. Subsequently the edges to the right of e_r are drawn analogously. Finally the vertices of the other levels and vertices on levels $i, i+1$ without edges in E_i are drawn. $\square$

In Case 2 of the proof it suffices to place vertices in the right quadrant to realize the implied intersection sequences because the danger of intersecting the wrong ray of a pseudoline is eliminated. This can in fact be generalized to arbitrary pseudoline arrangements whose pseudoline intersections lie on levels. Then vertices must only be placed in their cell of the arrangement.

Theorem 4.6. *Let $G = (V, E, \mathrm{lvl}, \prec)$ be a plane proper level graph and let $\mathcal{A}$ be a stretchable pseudoline arrangement with respect to G, such that each pseudoline intersection lies on a level of G. Furthermore, let A be a stretching of $\mathcal{A}$. Then there is a straight-line aligned drawing of (G, A).*

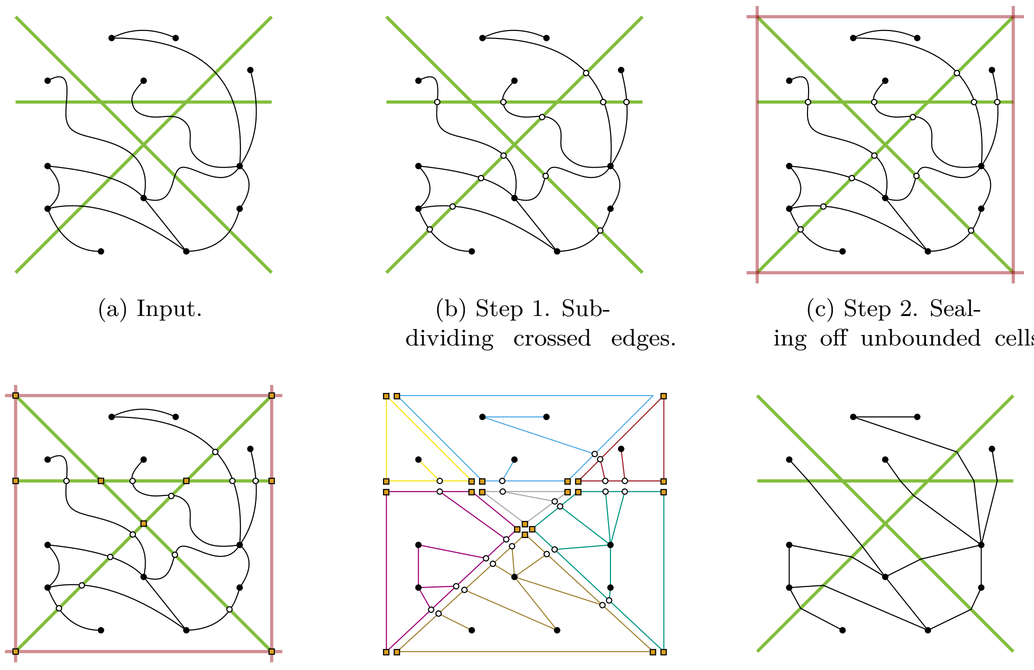
The proof is similar to Case 2 in the proof of Theorem 4.5. It boils down to placing each vertex v in the cell of A that corresponds to the cell of $\mathcal{A}$ that contains v with respect to $\prec$. The cell assignment ensures each stretching of a pseudoline intersects the same edges as the pseudoline. The order $\prec$ ensures that the order in which the edges are intersected is the same as on the pseudoline. Since G is proper and each pseudoline intersection lies on a level, it is predetermined which ray or segment between pseudoline intersections is intersected by an edge.

4.3 Aligned Level Drawings with Bends

Recall that $\zeta(e)$ is the number of pseudolines intersecting a non-aligned edge, or zero for an aligned edge. In this section, we prove the following theorem.

Theorem 4.7. *Let $G = (V, E, \mathrm{lvl}, \prec)$ be a plane level graph, let $\mathcal{A}$ be a pseudoline arrangement with respect to G such that $\mathcal{A} + \mathrm{lvl}(V)$ is stretchable and let $A + \mathrm{lvl}(V)$ be a stretching of $\mathcal{A} + \mathrm{lvl}(V)$. There is an aligned drawing of (G, A) such that every edge e is drawn with $\zeta(e)$ bends.*

Proof. Augment (G, A) according to the following steps.



(a) Input.

(b) Step 1. Subdividing crossed edges.

(c) Step 2. Sealing off unbounded cells.

(d) Step 3. Subdividing lines of A at intersections.

(e) Drawing each cell separately.

(f) Resulting Polyline Drawing.

Figure 4.5: Construction from Theorem 4.7. Note that levels are omitted in the drawing to reduce visual complexity.

Step 1: Substitute each intersection between a line in A and the interior of an edge with a dummy vertex

Step 2: Close unbounded cells of A with vertical/horizontal lines that lie beyond G. See the red vertical/horizontal lines in Figure 4.5(c). In the following, we ignore the new unbounded cells of A.

Step 3: Place a dummy vertex at every line-intersection in A.

Step 4: Add a dummy level for each of the dummy vertices. If a dummy vertex has a fixed y-coordinate (due to lying on a line-intersection) and other vertices share this y-coordinate, use the same level.

Step 5: Add edges on the boundaries of the cells of A such that for every cell the vertices on its boundary induce a cycle C in the order the vertices appear on the cell boundary.

For each cell, we want to draw the subgraph induced by it (interior and boundary) separately, using Theorem 2.5 by Eades et al. [EFLN06]. Let c be a cell and let $G[c]$ be the subgraph induced by c. We must triangulate $G[c]$ and make it st-plane. It already has a closed outer facial cycle C and a straight-line drawing P of C. First $G[c]$ is extended, using a source and sink elimination technique from Di Battista and Tamassia(Lemma 4.1 in [DBT88]), so that all sources and sinks lie on the top and bottom level. The elimination adds one upward outgoing edge to a sink at any but the top level and one incoming edge to a source at any but the bottom level. No vertices on C but the ones on the top level can be sinks and none but the ones on the bottom level can be sources. Therefore the source- and sink-elimination introduces no chords between vertices of C. This observation is important for the feasibility of P in the sense of the drawing algorithm (confer Definition 2.4).

If the bottom level contains one source, then no further modification is necessary. If it contains multiple sources, then the bottom level of $G[c]$ is due to a horizontal line closing the formerly unbounded c in Step 2. This means there are only two sources s_1, s_2. In this case we insert a supersource s on a level below the bottom level and connect it to the two sources. The augmentation modifies the outer facial cycle C and therefore we add another vertex s to P and connect it to the two sources. Observe that this modification does not introduce a chord between vertices on C and the modified P is convex if and only if the original P was convex. This is important for the feasibility of P in the sense of the drawing algorithm (confer Definition 2.4). Later, when a drawing is obtained, we remove s and include the straight line between s_1 and s_2 in the drawing instead. Multiple sinks on the top level are treated similarly. Therefore we can assume $G[c]$ is st-plane.

To triangulate $G[c]$ we add a vertex into each face and connect it to every vertex incident to that face, as also done by Eades et al. [EFLN06]. Observe that the triangulation step does not introduce chords between vertices on C. Again, this is important for the feasibility of P in the sense of the drawing algorithm (confer Definition 2.4), which brings us to the next point.

P is convex because its interior is the intersection of halfplanes induced by straight lines. Due to the observations that neither the source- and sink-eliminations nor the triangulation introduce chords between vertices of C, we only have the chords that were already present in $G[c]$. For any chord uv of C, there is a vertex due to a line-intersection of A between u and v in either of the two paths connecting u and v on C. That is because the subgraph induced by the vertices of G that are aligned by a single line in A is a linear forest (set of independent paths). Every vertex due to a line-intersection is drawn as an apex in P. Therefore P is feasible in the sense of Definition 2.4.

Using Theorem 2.5 we obtain a drawing Γ_c of $G[c]$. Combining the drawings of the cells yields a polyline drawing of (G, A) with i bends for every i-crossed edge, where each bend lies on a line of A. Aligned edges are drawn with zero bends. □

Corollary 4.8. *Let $G = (V, E, \text{lvl}, \prec)$ be a plane level graph and let $\mathcal{R}$ be a pseudoline with respect to G. There is a polyline drawing of $(G, \mathcal{R})$ where each edge crossed by $\mathcal{R}$ is bent at most once and free, and aligned edges are drawn without bends.*

Proof. Draw $\mathcal{R}$ as a vertical line R. Then apply Theorem 4.7. □

Da Lozzo et al. [DLDF+16] show a stronger result for general graphs and zero bends.

Theorem 4.9 (Da Lozzo et al. [DLDF+16]). *A plane graph $G = (V, E)$ has a planar straight-line drawing with $S \subset V$ drawn collinear if and only if there is a pseudoline with respect to G that passes through S.*

The proof of Theorem 4.7 consists of the first part of their proof, adapted to level graphs. They get rid of the bends, using a result of Pach and Tóth [PT04]. The latter finds an equivalent straight-line drawing with the same y-coordinates for any plane y-monotone drawing. Note that Da Lozzo et al. [DLDF+16] draw the pseudoline horizontally as opposed to vertically in our case. For level graphs the result of Pach and Tóth is not applicable as in order to keep vertices on vertical pseudolines we would need fixed x-coordinates in addition to levels' fixed y-coordinates.

Mchedlidze et al. [MRR17] give another proof for Theorem 4.9 that is based on triangulation, decomposition along chords and separating triangles as well as contraction of edges with a special property. Unfortunately, their result cannot be adapted in a straight-forward way because it is far from obvious how to contract edges in level graphs.

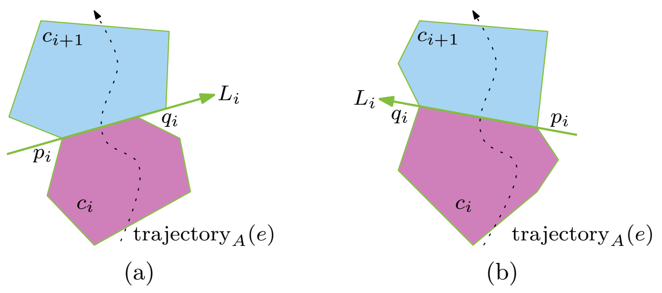


Figure 4.6: If cell c_i lies in the right halfplane of L_i, then trajectory$_A(e)$ crosses L_i to the right of p_i and to the left of q_i, as in (a). Conversely for c_i in the left halfplane of L_i as in (b)

4.4 Single Edge Intersection Preservation

Let $G = (V, E, \mathrm{lvl}, \prec)$ be a plane level graph, let $\mathcal{A}$ be a pseudoline arrangement with respect to G such that $\mathcal{A} + \mathrm{lvl}(V)$ is stretchable and let $A + \mathrm{lvl}(V)$ be a stretching of $\mathcal{A} + \mathrm{lvl}(V)$. In this section we want to look at straight-line aligned level drawings from the perspective of drawing G through the given stretching A, as opposed to drawing $\mathcal{A}$ through G. Let $e = uv \in E$ be a non-aligned edge. A straight-line drawing

$$\Gamma(e) = \left\{ \begin{pmatrix} X_u \\ \mathrm{lvl}(u) \end{pmatrix} + \lambda \begin{pmatrix} X_v - X_u \\ \mathrm{lvl}(v) - \mathrm{lvl}(u) \end{pmatrix} : \lambda \in [0,1] \right\}$$

of e is represented by the x-coordinates X_u, X_v for u and v.

The order in which e crosses pseudosegments of $\mathcal{A}$, pseudoline intersections of $\mathcal{A}$ and levels of G induces a y-monotone curve trajectory(e) in the cell complex of $\mathcal{A} + \mathrm{lvl}(V)$, see the black, dotted curve in Figure 4.7. The curve trajectory(e) starts at level-segment(u), ends at level-segment(v) and traverses cells of the cell complex of $\mathcal{A} + \mathrm{lvl}(V)$. The crossed pseudosegments, pseudoline intersections and levels are boundaries of the visited cells. Let trajectory$_A(e)$ be the curve through $A + \mathrm{lvl}(V)$, corresponding to trajectory(e) in $\mathcal{A} + \mathrm{lvl}(V)$. We call a drawing $\Gamma(e)$ through A *intersection-preserving* if and only if it intersects or visits exactly the same line segments and line intersections of A as well as levels as trajectory$_A(e)$ and does so in the same order. From this definition the following lemma follows immediately.

Lemma 4.10. *A straight-line drawing Γ of G is an aligned level drawing (Γ, A) of $(G, \mathcal{A})$ if and only if the edge drawing of every non-aligned edge is intersection-preserving and every aligned edge is drawn on the line corresponding to the pseudoline that aligns it.*

We now introduce the notion of canal polygons, a tool to formulate intersection preservation for single non-aligned edges as a visibility problem.

4.4.1 Canal Polygon

We distinguish two cases; whether trajectory(e) visits a pseudoline intersection or whether it does not. For the first case we refer to the next paragraph *Non-Simple Canal Polygons* and assume for this paragraph that trajectory(e) does not visit a pseudoline intersection.

Let $c_1, \dots, c_h$ be the sequence of cells in cells$(A + \mathrm{lvl}(V))$ visited by trajectory$_A(e)$. Furthermore let $s_0, \dots, s_h$ be the sequence of line segments and level-segments of vertices and level-crossings, traversed by trajectory$_A(e)$. Herein we have s_o = level-segment(u) and s_h = level-segment(v). The cell c_i has s_{i-1} and s_i on its boundary. Let L_i be the

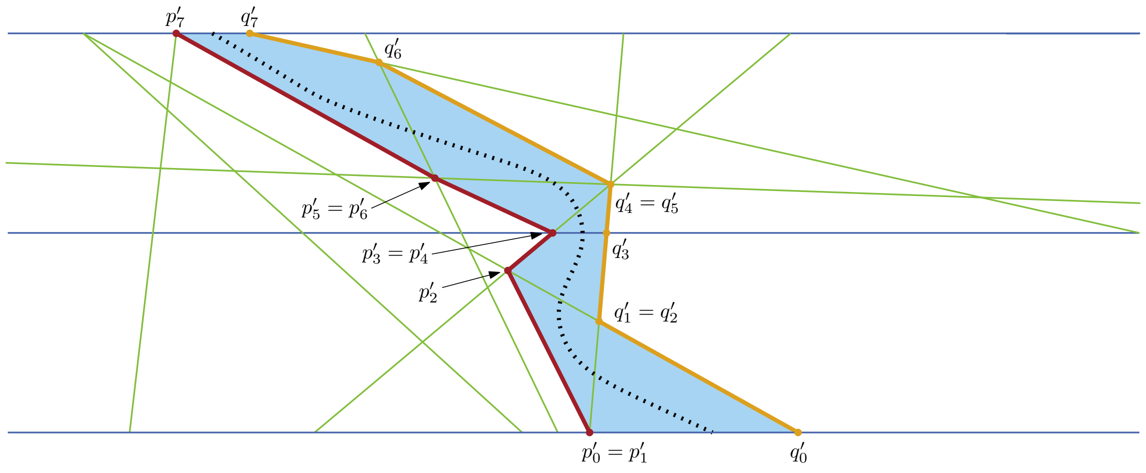


Figure 4.7: Line arrangement A in green, levels in dark blue. The dotted black curve represents $\text{trajectory}_A(e)$ and the interior of the canal polygon $\text{CP}(e)$ is drawn in light-blue. Its left boundary $\text{CP}_l(e)$ is drawin in red, its right boundary $\text{CP}_r(e)$ in orange.

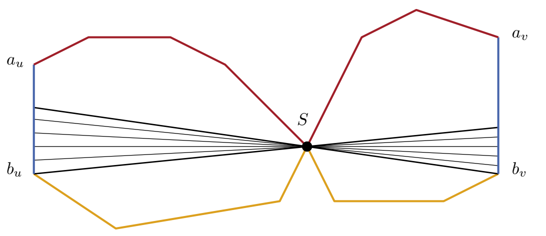
line or level, from which s_i stems. Recall that pseudolines (and therefore lines in A) are by definition oriented upwards and we orient levels from left to right. Let $s_i = (p_i, q_i)$, where p_i comes before q_i in the orientation of L_i. See Figure 4.6 for an illustration of this scenario. If c_i lies in the right halfplane of L_i, then $\text{trajectory}_A(e)$ crosses s_i with p_i to its left and q_i to its right; conversely if c_i lies in the left halfplane. Let p'_i be p_i if c_i lies in the right halfplane of L_i and q_i otherwise. Similarly let q'_i be q_i if c_i lies in the right halfplane and p_i otherwise. Therefore $\text{trajectory}_A(e)$ crosses s_i with p'_i to its left.

We define the *canal polygon* $\text{CP}(e)$ of e as the polygon with the vertices $p'_0, \ldots, p'_h, q'_h, \ldots q'_0, p'_0$. It has a *left boundary* $\text{CP}_l(e) = p'_0, \ldots p'_h$ and a separate *right boundary* $\text{CP}_r(e) = q'_0, \ldots, q'_h$. Furthermore it is horizontally bounded by level-segment(u) and level-segment(v). Figure 4.7 shows a canal polygon, with the left boundary highlighted in red and the right boundary in orange. Note that it lies inside the chain of cells induced by the pseudolines (green). Its left boundary has one edge per cell c_i, joining the two "left" points p'_{i-1} and p'_i on the entry segment s_{i-1} and exit segment s_i of $\text{trajectory}_A(e)$ in cell c_i.

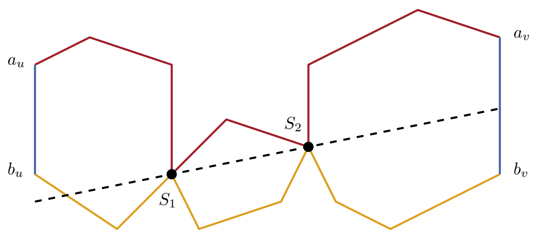
Intuitively, the canal polygon is the canal through which e has to be drawn. It lies in the geometric figure obtained by chaining $c_1, \ldots, c_h$ together at their shared boundary segments. Since every cell c_i is convex and $\text{trajectory}(e)$ does not visit a pseudoline intersection, the canal polygon $\text{CP}(e)$ is non-self-intersecting, i.e. it is a simple polygon. The x-coordinates X_u, X_v of u, v in an intersection-preserving drawing $\Gamma(e)$ of e must be chosen in such a way, that the line segment between $(X_u, \text{lvl}(u))$ and $(X_v, \text{lvl}(v))$ intersects neither the left nor right boundary of $\text{CP}(e)$. We formulate this in the following lemma.

Lemma 4.11. *Let $e = uv \in E$ with a simple canal polygon $\text{CP}(e)$. A straight-line drawing $\Gamma(e)$ of e is intersection-preserving if and only if its interior does not intersect the left or right boundary of $\text{CP}(e)$ and the endpoints of $\Gamma(e)$ are on* level-segment(u), level-segment(v).

This property is related to visibility in polygons (recall Section 2.5). The endpoints of every intersection-preserving drawing are visible from another in $\text{CP}(e)$. Conversely a line segment joining a point on level-segment(u) and a point on level-segment(v) that are visible from another in $\text{CP}(e)$ is an intersection-preserving drawing if and only if it does not intersect $\text{CP}_l(e)$ or $\text{CP}_r(e)$ in a vertex of $\text{CP}(e) \setminus \{\text{level-segment}(u), \text{level-segment}(v)\}$. Let p be a point on level-segment(u) and let q_a be the lefmost point, and let q_b be the rightmost point, on level-segment(v), visible from p. Then the closed interval $[q_a, q_b]$ is



(a) Non-simple canal polygon with one cutvertex. The straight-line intersection-preserving drawings of e are the line segments through the cutvertex with endpoints on level-segment$(u) = (a_u, b_u)$ and level-segment$(v) = (a_v, b_v)$, which do not intersect $\mathrm{CP}_l(e), \mathrm{CP}_r(e)$.



(b) Non-simple canal polygon with two cutvertices. The black dashed line segment passes through S_1 and S_2. It is the only candidate for a straight-line intersection-preserving drawing. Since it does not end on level-segment$(u) = (a_u, b_u)$ and intersects $\mathrm{CP}_r(e)$ there is no straight-line intersection preserving drawing of e.

Figure 4.8: Non-simple canal polygons. Both rotated by 90 degrees clockwise.

visible from p and the open interval (q_a, q_b) is the set of endpoints of intersection-preserving straight-line drawings of e in which vertex u is drawn as p.

Non-Simple Canal Polygons

We now assume that trajectory(e) visits a pseudoline intersection. If we define a canal polygon analogously to the previous paragraph, its left and right boundary meet at the corresponding line intersection of $A + \mathrm{lvl}(V)$, making the polygon self-intersecting, i.e. *non-simple*. We distinguish whether trajectory(e) visits one pseudoline intersection or two and more. See Figure 4.8 for the distinction of these two cases.

If trajectory$_A(e)$ visits exactly one line intersection S of A (see Figure 4.8(a)) the intersection-preserving straight-line drawings of e are exactly those line segments $\overline{p_u p_v}$ through S with p_u on level-segment(u), p_v on level-segment(v) and the two pieces of $\overline{p_u p_v} \setminus \{S\}$ do not intersect the boundary of the canal polygon. In Figure 4.8(a) we see the straight-line intersection-preserving drawings. In particular the edge drawing at b_u is the one with the leftmost visible point on level-segment(v) and the edge drawing at b_v is the one with the leftmost visible point on level-segment(u).

If trajectory$_A(e)$ visits two or more line intersections $S_1, S_2, \ldots, S_j$ (see Figure 4.8(b)) there is at most one intersection-preserving straight-line drawing since S_1 and S_2 must be aligned by it. If $S_1, S_2, \ldots$ are not collinear, there is no straight-line intersection-preserving drawing. Otherwise the one remaining candidate $\overline{p_u p_v}$ with p_u on level-segment(u), p_v on level-segment(v), which passes through every S_i is a straight-line intersection-preserving drawing if and only if every piece of $\overline{p_u p_v} \setminus (\{S_i | i = 1, \ldots, j\} \cup \{p_u, p_v\})$ does not intersect the boundary of the canal polygon.

We conclude this section with the following lemma on the size of canal polygons. Recall that $\zeta(e)$ is the number of pseudolines, which intersect edge e or 0 if e is an aligned edge, and recall that span$(e) = \mathrm{lvl}(v) - \mathrm{lvl}(u)$, for $e = uv$. Furthermore recall that k is the number of levels and z is the number of pseudolines in $\mathcal{A}$.

Lemma 4.12. *$|\mathrm{CP}(e)| \in O(\mathrm{span}(e) + \zeta(e)) \subset O(k + z)$.*

Proof. Every pseudosegment and level-segment crossed by trajectory(e) contributes at most two vertices to CP(e). □

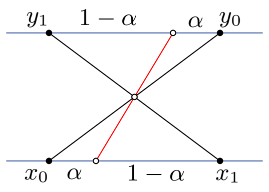


Figure 4.9: Illustration of the claim in the case l_0 and l_1 intersect in the proof of Lemma 4.14. l_α in red, l_0, l_1 in black. For any $\alpha \in [0,1]$, the line segment l_α associated with the convex combination $(1-\alpha)p_0 + \alpha p_1$ runs through the intersection of l_0 and l_1.

4.4.2 Edge Drawing Polygon

In this section we describe a polygon-based, explicit representation for all straight-line intersection-preserving drawings of $e = uv \in E$, namely the *edge drawing polygon* EDP(e). We use it in Chapter 5 to efficiently decide Straight-Line Aligned Level Drawing in graphs with two levels.

Define $Q(e)$ as the geometric figure $Q(e) := \{(X_u, X_v)\} \subset \mathbb{R}^2$ such that X_u, X_v are the x-coordinates of the respective two endpoints on level-segment(u), level-segment(v) of a straight-line intersection-preserving drawing $\Gamma(e)$ of e. If CP(e) is simple it follows from the definition of intersection-preserving drawings that $Q(e)$ is an open set. If CP(e) is non-simple, $Q(e)$ is either a line segment, single point or empty. As the representation and computation of intersection-preserving drawings in non-simple canal polygons is easy, we will focus on simple canal polygons.

So assume for the remainder of this section that CP(e) is simple. We define the *edge drawing polygon* EDP(e) as the boundary of $Q(e)$. To justify the name edge drawing polygon we should argue that it is in fact a polygon; even a convex polygon. We do so by establishing a connection to weak edge visibility (recall Section 2.5). First we prove that $Q(e)$ is convex and use the convexity to subsequently argue that EDP(e) is a polygon.

Lemma 4.13. *If* CP(e) *is simple then* $Q(e)$ *is convex.*

Proof. Let $p_0 = (x_0, y_0), p_1 = (x_1, y_1)$ be two points in the interior of $Q(e)$. We show that any convex combination $(1-\alpha)p_0 + \alpha p_1$ of p_0 and p_1 with $\alpha \in [0,1]$ lies in $Q(e)$.

Let

$$l_\alpha := \left\{ \begin{pmatrix} (1-\alpha)x_0 + \alpha x_1 \\ \mathrm{lvl}(u) \end{pmatrix} + \lambda \begin{pmatrix} (1-\alpha)(y_0 - x_0) + \alpha(y_1 - x_1) \\ \mathrm{lvl}(v) - \mathrm{lvl}(u) \end{pmatrix} \mid \lambda \in [0,1] \right\}$$

be the line segment (potential edge drawing) associated with the convex combination $(1-\alpha)p_0 + \alpha p_1$, e.g. l_0 is the drawing associated with p_0.

Assume $x_0 < x_1$ without loss of generality. If $y_0 < y_1$ then for $\alpha \in [0,1]$ we have that l_α lies between x_0 and x_1 on level-segment(u) and between y_0 and y_1 on level-segment(v). Therefore $\begin{pmatrix} (1-\alpha)x_0 + \alpha x_1 \\ \mathrm{lvl}(u) \end{pmatrix}$ and $\begin{pmatrix} (1-\alpha)y_0 + \alpha y_1 \\ \mathrm{lvl}(v) \end{pmatrix}$ are visible from another.

If $y_1 < y_0$ the line segments l_0 and l_1 intersect in a point $l_0 \cap l_1$.

Claim. For any fixed $\alpha \in [0,1]$ the line segment l_α intersects l_0 and l_1 in $l_0 \cap l_1$, see Figure 4.9. From the claim it follows immediately that the endpoints of any l_α are visible from another. We now prove the claim.

Let λ_0, λ_1 be the parameters of l_0, l_1 at their respective intersection with l_α and let λ_α be the parameter of l_α at the intersection of l_0 and l_α. For any $\beta \in [0,1]$, the y-coordinate $\mathrm{lvl}(u) + \lambda(\mathrm{lvl}(v) - \mathrm{lvl}(u))$ of l_β is independent of β and therefore $\lambda_0 = \lambda_1 = \lambda_\alpha$. From

$$x_0 + \lambda_0(y_0 - x_0) = (1-\alpha)x_0 + \alpha x_1 + \lambda_\alpha[(1-\alpha)(y_0 - x_0) + \alpha(y_1 - x_1)]$$

and $\lambda_\alpha = \lambda_0$ we conclude

$$\lambda_0 = \frac{x_0 - x_1}{y_1 - x_1 - (y_0 - x_0)} \,. \tag{4.1}$$

Substitute $\lambda_1 = \lambda_0$ by Equation 4.1, in the equation for the x-coordinate of l_1:

$$x_1 + \lambda_1(y_1 - x_1) = x_1 + \frac{x_0 - x_1}{y_1 - x_1 - (y_0 - x_0)}(y_1 - x_1) = \frac{x_0 y_1 - x_1 y_0}{y_1 - x_1 - (y_0 - x_0)} \,.$$

Similarly substitute λ_0 by Equation 4.1 in the equation for the x-coordinate of l_0:

$$x_0 + \lambda_0(y_0 - x_0) = x_0 + \frac{x_0 - x_1}{y_1 - x_1 - (y_0 - x_0)}(y_0 - x_0) = \frac{x_0 y_1 - x_1 y_0}{y_1 - x_1 - (y_0 - x_0)} \,.$$

From the equality of the previous two equations, we conclude that the x-coordinate of the intersection of l_0 and l_1 as well as the x-coordinate of the intersection of l_0 and l_α are equal. □

Now we argue that EDP(e) is a polygon by explicitly stating its vertices and edges. First we recall the correspondence between points in $Q(e)$ and line segments joining points on level-segment(u) and level-segment(v), then we recall weak edge visibility and finally establish the connection.

Let (X_u, X_v) be a point on the boundary of $Q(e)$. It corresponds to a straight line segment

$$l(X_u, X_v) := \left\{ \begin{pmatrix} X_u \\ \mathrm{lvl}(u) \end{pmatrix} + \lambda \begin{pmatrix} X_v - X_u \\ \mathrm{lvl}(v) - \mathrm{lvl}(u) \end{pmatrix} \mid \lambda \in [0,1] \right\}$$

through CP(e) with two endpoints $(X_u, \mathrm{lvl}(u))^{\mathrm{T}}$ and $(X_v, \mathrm{lvl}(v))^{\mathrm{T}}$ which are visible from another in CP(e). Since (X_u, X_v) lies on the boundary of $Q(e)$, the line segment $l(X_u, X_v)$ intersects the left or right boundary of CP(e) in a vertex.

Now we recall the weak edge visibility algorithm of O'Rourke [O'R87] as described in Section 2.5 for left critical lines. Right critical lines are analogous. Let s be the polygon-edge from which weak visibility is computed in polygon P, let W be a visibility window and let H be the sweepline. Denote by $L(x)$ the line segment connecting a point x with its leftmost visible point on s, where x is a point on H in visibility window W. The point of contact $c[L(x)]$ is defined as the closest vertex on the left boundary of the visibility window. W is partitioned into intervals such that the point of contact $c[L(x)]$ does not change as x varies over the interval. The points of contact on the left boundary form a convex chain. A point x where $c[L(x)]$ does change induces a left critical line $L_i = L(x)$. Observe that any left critical line either intersects two consecutive vertices on the convex chain of contact points (such as L_3, L_2, L_1 in Figure 4.10) or x is the rightmost point in W that is visible from s (such as L_0 in Figure 4.10).

EDP(e) can be extracted from the critical lines in the last step of the algorithm, when we compute weak edge visibility from the edge level-segment(u) in CP(e). In the last sweepline advancement step of the algorithm, when H is at the y-coordinate lvl(v), the left critical lines are those line segments that correspond to vertices on the left hull of EDP(e).

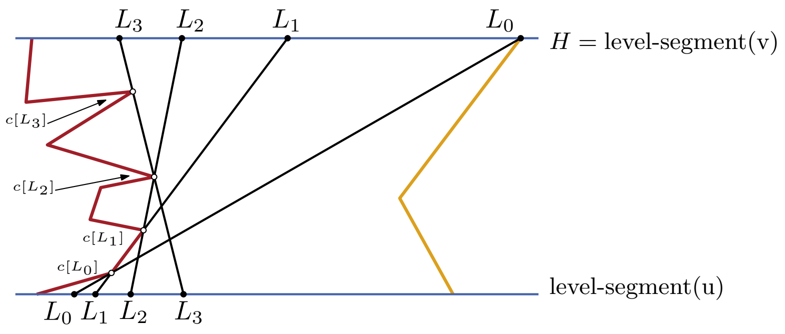


Figure 4.10: Left critical lines when computing weak edge visibility from level-segment(u) in CP(e).

Denote by X_u the x-coordinate of the line segment's endpoint on level-segment(u) and by X_v the x-coordinate of the endpoint on level-segment(v). Then (X_u, X_v) is a point on the boundary of $Q(e)$ because $(X_u, \mathrm{lvl}(u))^{\mathrm{T}}, (X_v, \mathrm{lvl}(v))^{\mathrm{T}}$ are visible from another in CP(e) via the critical line and the critical line intersects the left boundary of CP(e). Since the critical line intersects the left boundary even in two points or connects to the rightmost point on level-segment(v) visible from level-segment(u), the point (X_u, X_v) is a vertex of EDP(e). In particular the latter case yields the topmost vertex of the left hull. The line segments $L(X_v)$, as X_v varies over an interval in which the point of contact $c[L(X_v)]$ does not change, correspond to points on the boundary of $Q(e)$ which lie on an edge of EDP(e). Similarly the right critical lines correspond to vertices on the right hull of EDP(e). This yields the following lemma.

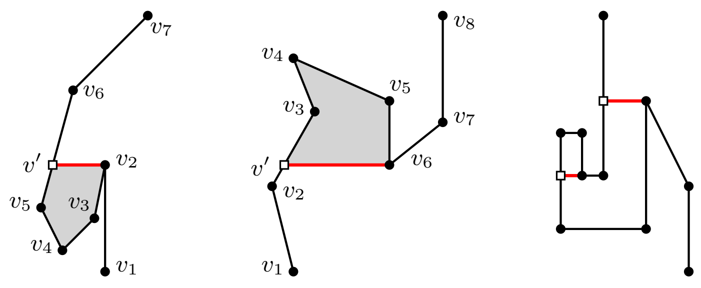
Lemma 4.14. *If* CP(e) *is simple then* EDP(e) *is a convex polygon and its size is bounded by the size of the canal polygon:* $|\,\mathrm{EDP}(e)| \leq |\,\mathrm{CP}(e)|$.

Computing Edge Drawing Polygons

We could immediately use the weak edge visibility algorithm to compute EDP(e) in $O(|\,\mathrm{CP}(e)| \log |\,\mathrm{CP}(e)|)$ time. However, it turns out that with some modifications $O(|\,\mathrm{CP}(e)|)$ running time is possible. There are three parts of the algorithm, which contribute to the logarithmic factor in the running time. The polygon-vertices are sorted by y-coordinate and the use of balanced search trees to maintain the edges intersected by H as well as the critical lines. In the following we discuss how to perform the run-time critical parts in linear time for canal polygons, particularly for the left boundary. The right boundary is treated similarly.

Pocket Elimination

The boundary of the canal polygon consists of two separate vertex chains, one left boundary, one right boundary, connecting the left, respectively right ends of level-segment(u) and level-segment(v). Let $v_1, v_2, v_3, \ldots$ denote the vertex chain of the left boundary and denote by $y(v_i)$ their y-coordinates, where $y(v_1) = \mathrm{lvl}(u)$. A subchain $v_{i-1}, v_i, v_{i+1}, \ldots, v_j$ is called an *above-pocket* if the chain exhibits a left turn at v_i, if $y(v_i) > y(v_{i+1})$, $y(v_{i-1}) < y(v_i)$



(a) An above-pocket $v_1, v_2, v_3, \ldots, v_6$ (b) A below-pocket $v_2, \ldots, v_4, \ldots, v_6$ (c) Nested pockets

Figure 4.11: Pockets (gray) in a canal polygon. The horizontal line segments (red) eliminate the shown pockets.

and v_j is the first vertex after v_i with $y(v_j) \geq y(v_i)$. See Figure 4.11(a) for an illustration. For $i = 2$, there is an above-pocket $v_1, v_2, v_3, \ldots, v_6$.

Similarly a *below-pocket* is a subchain $v_i, \ldots, v_j, \ldots v_o$ which exhibits a right turn at v_j, a left turn at v_o and has $y(v_i) < y(v_{i+1})$, $y(v_{j-1}) < y(v_j) > y(v_{j+1})$, $y(v_{o-1}) > y(v_o) < y(v_{o+1})$ and finally $y(v_i) \leq y(v_o) \leq y(v_{i+1})$. See Figure 4.11(b) for a below-pocket with $i = 2, j = 4, o = 6$.

Pockets are the non-y-monotonic subchains of the left boundary. A below-pocket is not visible from level-segment(v) and an above-pocket is not visible from level-segment(u). Hence, we can eliminate pockets from the canal polygon without losing edge drawings.

An above-pocket $v_{i-1}, v_i, v_{i+1}, \ldots, v_j$ is eliminated by placing a dummy vertex v' (white square in Figure 4.11) on $v_{j-1}v_j$ with $y(v') = y(v_i)$. The new subchain of the left boundary then consists of v_{i-1}, v_i, v', v_j (red horizontal line), for example v_1, v_2, v', v_6 in Figure 4.11(a). The elimination of above-pockets obviously takes linear time in the subchain size. A below-pocket $v_i, \ldots, v_j, \ldots, v_o$ is eliminated by placing a dummy vertex v' on v_iv_{i+1} with $y(v') = y(v_o)$. The new subchain of the left boundary then consists of v_i, v', v_o. The elimination of below-pockets takes linear time in the subchain size. As Figure 4.11(c) indicates, pockets can be nested. In every nesting there is always an outmost pocket whose elimination covers its inner pockets.

The newly obtained boundaries are sorted by y-coordinates. By merging the two sorted boundary vertex lists we obtain a vertex list sorted by y-coordinates in linear time.

Horizontal Polygon Edges

Pocket-elimination introduces horizontal edges to the boundaries. The other source of horizontal edges are levels as part of the boundary. Whereas the unmodified weak edge visibility algorithm assumes all edges (but the starting edge) are not horizontal, we need to incorporate them. Let $v_1, \ldots v_n$ be the vertices of the left boundary of the pocket-eliminated canal polygon. In a sweepline traversal of the pocket-eliminated canal polygon, let v_i be the next vertex. The treatment of horizontal edges replaces the old Case 3 of window updates due to v_i, which dealt with splitting windows.

Furthermore, the pocket-elimination eliminates all visibility windows but the one, through which e runs. Thus the distribution of critical lines between windows is unnecessary and the then inactive windows can be deleted.

Case 3'. The outgoing edge v_iv_{i+1} of v_i is horizontal and the left boundary has a left turn at v_i.

This case corresponds to an eliminated above-pocket or a level on the left boundary. Let v_i be located between R_j and R_{j+1} as well as between L_{h+1} and L_h. Then $R_0, \dots, R_j$ and $\dots, L_{h+2}, L_{h+1}$ are deleted. Finally $L(v_i)$ and $R(v_i)$ are inserted into L and R.

Case 3". The incoming edge $v_{i-1}v_i$ of v_i is horizontal and the left boundary has a right turn at v_{i-1}.

This case corresponds to an eliminated below-pocket and is similar to Case 3'. Let v_i be located between R_j and R_{j+1} as well as between L_{h+1} and L_h. Then $R_{j+1}, R_{j+2}, \dots$ and $L_h, \dots, L_0$ are deleted and $L(v_i)$ and $R(v_i)$ are inserted into L and R.

Maintaining Windows and Critical Lines

As already mentioned, we only need to maintain the single window W through which e runs, i.e. no search-tree is necessary. Maintaining critical lines does not require a search tree either and is instead done with a doubly-linked list each for L and R. The traversal direction is, as always, for vertices on the left boundary; the right boundary is treated similarly. Locating v_i in Case 3' is done by traversing L and R in left-to-right order on H. The linear work to traverse the lists up to v_i is mitigated by the fact that all critical lines left of v_i are deleted, i.e. are traversed at most once in a localization step for Case 3'. Additionally the insertion of $L(v_i), R(v_i)$ takes constant time, as they are leftmost on H. Similarly in Case 3", v_i is located by traversing the lists in right-to-left order on H, again yielding constant time per inserted and deleted critical line. The other cases of window updates due to v_i do not entail updates to L, R. Locating the leftmost visible point x_a on H for window updates due to advancing H is done by traversing L, R from left to right on H. Again, the critical lines to the left of x_a are deleted and thus only constant time is spent per critical line.

In conclusion, we have shown how to reduce the runtime of all three components of the weak edge visibility algorithm that introduce a logarithmic runtime factor down to linear time. With these modifications we can compute $\mathrm{EDP}(e)$ in $O(|\,\mathrm{CP}(e)|)$ time.

Lemma 4.15. *The edge drawing polygon* $\mathrm{EDP}(e)$ *of* e *can be computed in* $O(|\,\mathrm{CP}(e)|)$ *time.*

4.5 Linear Constraints for Intersection Preservation

In this section we present a set of linear constraints to obtain an intersection-preserving drawing. We build upon $\mathcal{LCP}$-FE-SL from Equation 3.5 in Section 3.1 as the foundation.

Recall for edge $e = (s,t) \in E$, the definition of the x-slope of its drawing $M_e = \dfrac{X_t - X_s}{y_t - y_s}$. We can evaluate the drawing's x-value for a fixed y-coordinate $y \in [y_s, y_t]$ as $x = M_e \cdot (y - y_s) + X_s$.

Let $e = (s,t)$ be a non-aligned edge with simple canal polygon. By Lemma 4.11 a drawing of e induced by (X_s, X_t) is intersection-preserving, if and only if its interior intersects neither the left nor right boundary of $\mathrm{CP}(e)$ and $X_s \in \text{level-segment}(s), X_t \in \text{level-segment}(t)$. Since a straight line can intersect a polygon-edge at most once, it suffices to only check at the vertices of $\mathrm{CP}(e)$, whether $\overline{X_sX_t}$ lies in the interior. More precisely, for a vertex $p = (x_p, y_p)$ of the left boundary $\mathrm{CP}_l(e)$, we verify whether $\overline{X_sX_t}$ lies to the right of x_p at height y_p.

The drawing of a non-aligned edge e with non-simple canal polygon must additionally visit the line intersections of A that are visited by $\text{trajectory}_A(e)$. We denote these line intersections by $\mathrm{CP}_{\chi(A)}(e)$ and do not view them as part of the left or right boundary $\mathrm{CP}_l(e), \mathrm{CP}_r(e)$. For edges with simple canal polygon the set $\mathrm{CP}_{\chi(A)}(e)$ is empty. The

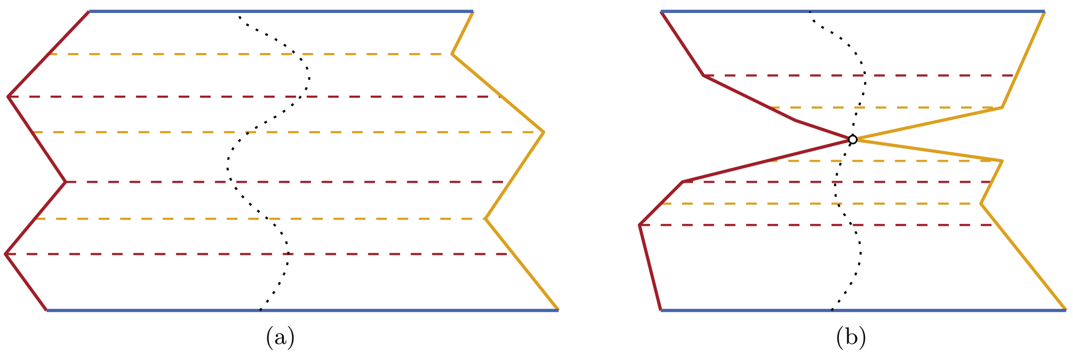


Figure 4.12: Left: linear constraints in a simple canal polygons. Red dashed constraints are due to the left boundary, orange constraints due to the right boundary. Right: linear constraints in a non-simple canal polygon (white disk).

following equation constitutes the constraints for an intersection-preserving drawing of a non-aligned edge e.

$$\begin{aligned}\text{IntersectionPreservationOneEdge}(e = (s,t)) = \\ &\bigcup_{(x_p,y_p)\in \mathrm{CP}_l(e)} \{M_e \cdot (y_p - y_s) + X_s > x_p\} \\ \cup &\bigcup_{(x_q,y_q)\in \mathrm{CP}_r(e)} \{M_e \cdot (y_q - y_s) + X_s < x_q\} \\ \cup &\bigcup_{(x_o,y_o)\in \mathrm{CP}_{\chi(A)}(e)} \{M_e \cdot (y_o - y_s) + X_s = x_o\}\end{aligned} \tag{4.2}$$

See Figure 4.12 for an illustration of the constraints. An aligned edge $e = uv$ has a fixed drawing. We can either replace all occurences of X_u, X_v with constants or introduce constraints that fix X_u, X_v. By combining the constraints introduced in this section with the constraints for straight-line drawings (Equation 3.3) and the embedding (Equation 3.1) in one linear constraint program $\mathcal{LCP}$-AD we can solve Straight-Line Aligned Level Drawing instances. From the running time of Kamarkar's algorithm (Theorem 2.12) we obtain the following lemma.

Lemma 4.16. *The running time of the linear constraint program $\mathcal{LCP}$-AD is*

$$O([|V| + \sum_{e\in E} |\operatorname{CP}(e)|]^{3.5} + L^2)$$

Here L is the number of bits in the input to the linear program. We now analyze the factor L. To that end, we assume that the streched line arrangement A is given as the set of points where lines intersect. Furthermore, we assume that the representation of such a point requires $2w$ bits, w bits for every coordinate.

Define l as the number of level-crossings. Every constraint in Equation 4.2 requires $O(\log(l + |V|))$ bits to identify the two involved variables X_s, X_t. The coefficients require $O(w + \log k)$ bits as the denominator stemming from M_e has $\binom{k}{2}$ possible values, the terms y_q, y_p, y_o require w bits each and the rest requires a constant number of bits. The right side of every constraint in Equation 4.2 requires w bits.

By combining this with Lemma 3.3 we obtain Equation 4.3.

$$L \in O\left(\log(l + |V|) \cdot \left[\sum_{e\in E} |\operatorname{CP}(e)|(w + \log k) + (l + |E|)\log k\right]\right) \tag{4.3}$$

From Lemma 4.12 we know that the size of an edge's canal polygon is linear in the span of the edge and the number of pseudolines crossing the edge. Therefore we obtain the following theorem.

Theorem 4.17. *There is a polynomial time algorithm to decide* STRAIGHT-LINE ALIGNED LEVEL DRAWING*.*

5. Monotonic Polygon Hitting Set

In this chapter we discuss the Monotonic Polygon Hitting Set problem, a geometric problem that solves Straight-Line Aligned Level Drawing when the graph has two levels. We start with basic definitions and the problem statement. Subsequently Section 5.1 outlines the algorithmic approach. In Section 5.2 and Section 5.3 we introduce elementary operations, which are used in Section 5.4, where we propose an asymptotically optimal algorithm to solve Monotonic Polygon Hitting Set (Corollary 5.13). The reduction from Straight-Line Aligned Level Drawing with two-level graphs to Monotonic Polygon Hitting Set is described in Section 5.5.

Definition 5.1 (Polygon Hitting Set)**.** *For a set of m convex polygons $\{P_1, \dots, P_m\}$, a* hitting set *is a set $\{p_1, \dots, p_m\}$ of m points s.t. p_i lies in the interior of P_i.*

Definition 5.2 (Point Ordering)**.** *By $<_{xy}$ we denote the relation between two points $p_1 = (x_1, y_1), p_2 = (x_2, y_2)$ that is fulfilled exactly if $x_1 < x_2$ and $y_1 < y_2$. By $<_x$ we denote the relation $x_1 < x_2$ and $y_1 = y_2$ as well as analogously $<_y$ for $x_1 = x_2$ and $y_1 < y_2$.*

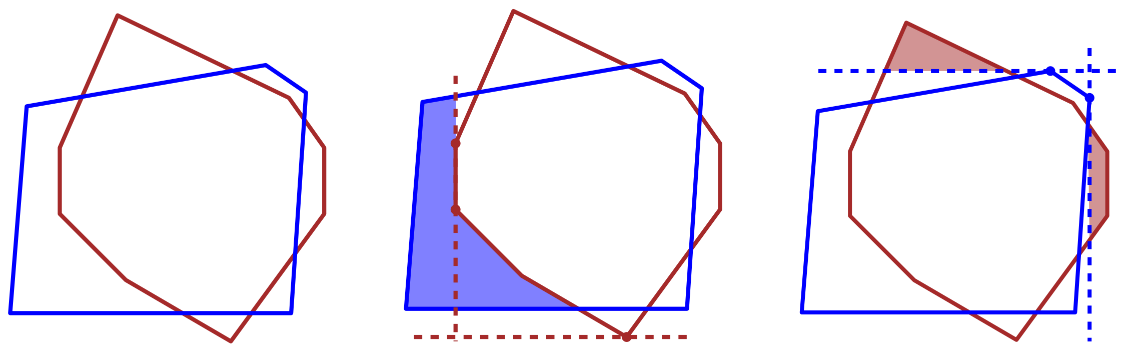
A sequence of points that is ordered using a combination of these three comparison operators is totally ordered. In a sequence of convex polygons we can compare two consecutive polygons P_i, P_{i+1} *symbolically* by a comparison operator $\lhd_i \in \{<_{xy}, <_x, <_y\}$. Notationwise, we write $P_i \lhd_i P_{i+1}$ for the symbolic comparison of polygons. Regarding a hitting set of the ordered polygons, we can ask for a sequence of points (each point in the interior of its respective polygon) that geometrically sustains the same relations as the polygon sequence does symbolically.

Definition 5.3. *Let $\mathcal{P} = \langle P_1, \dots, P_m \rangle$ be a sequence of convex polygons. Let $\mathcal{S} = \langle \lhd_1, \dots, \lhd_{m-1} \rangle$ be a sequence of comparisons with $\lhd_i \in \{<_x, <_y, <_{xy}\}$. A hitting set $p_1, \dots, p_m$ of $\mathcal{P}$ is $\mathcal{S}$-*monotonic *if and only if $p_i \lhd_i p_{i+1}$.*

Definition 5.4. *Let $\mathcal{P}, \mathcal{S}$ be as in Definition 5.3.* Monotonic Polygon Hitting Set *is the problem of deciding whether $\mathcal{P}$ has a $\mathcal{S}$-monotonic hitting set.*

5.1 Outline

Our approach to obtaining a $\mathcal{S}$-monotonic hitting set is based on removing those parts of the polygons that cannot be in a $\mathcal{S}$-monotonic hitting set, called *infeasible regions.* We show that this suffices to obtain a solution by a simple iterative traversal.



(a) Two convex polygons P in brown and Q in blue.

(b) $<_{xy}$-infeasible regions of Q regarding P.

(c) $<_{xy}$-infeasible regions of P regarding Q.

Figure 5.1

First we discuss the *infeasible regions* of a polygon. We start with infeasibility in a two-polygon scenario $P \lhd Q$ and a way to compute the infeasible regions. This is discussed in detail for the $P <_{xy} Q$ comparison operator as there are only small changes necessary for the $<_x$ and $<_y$ comparators. Then we proceed to extend this result to an arbitrary-length sequence $\mathcal{P} = \langle P_1, \ldots, P_m \rangle$ by considering infeasibility regarding all prefixes of $\mathcal{P}$, i.e. the set $\{\langle P_1, \ldots, P_j \rangle | 1 \leq j \leq m\}$. This immediately yields a linear time algorithm for computing a $\mathcal{S}$-monotonic hitting set of $\mathcal{P}$ as shown in Corollary 5.13 of Theorem 5.12 in Section 5.4.

5.2 Infeasible Regions for $P <_{xy} Q$

In this section we assume the scenario of two convex polygons P, Q ordered by $P <_{xy} Q$ and show how to compute a $<_{xy}$-monotonic hitting set $p <_{xy} q$ or decide there is none. Figure 5.1 shows that scenario. By default we assume P and Q to be non-empty.

Definition 5.5 ($<_{xy}$-infeasibility regarding P)**.** *We call a point $q = (x_q, y_q)$ in the interior of Q $<_{xy}$-*infeasible* regarding P, if there is no point $p = (x_p, y_p)$ in the interior of P with $x_p < x_q$ and $y_p < y_q$.*

By definition a $<_{xy}$-infeasible point of Q regarding P cannot be in a $<_{xy}$-monotonic hitting set. Conversely if Q has no $<_{xy}$-infeasible points regarding P, any point of Q is in at least one $<_{xy}$-monotonic hitting set. By $\widehat{Q}$ we denote the maximal subpolygon of Q (interior($\widehat{Q}$) $\subset$ interior(Q)) with no $<_{xy}$-infeasible points regarding P.

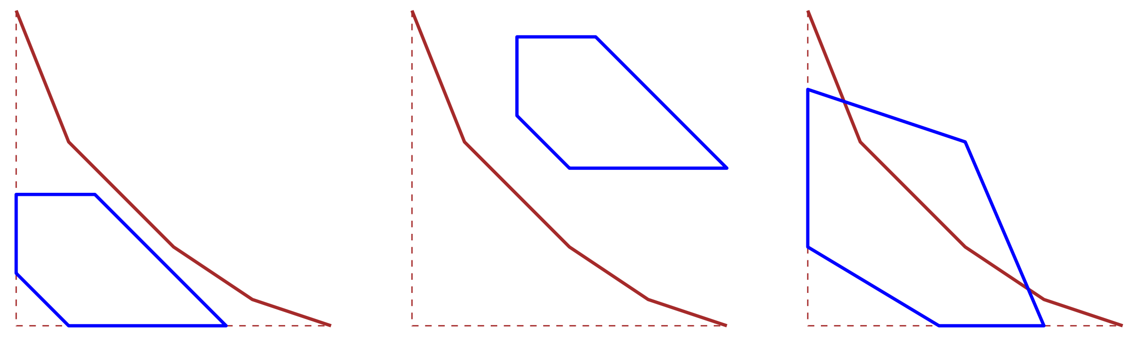
Lemma 5.6. *For every point $q \in$ interior($\widehat{Q}$) there is a point $p \in$ interior(P) such that $p <_{xy} q$ is a $<_{xy}$-monotonic hitting set of $P <_{xy} Q$.*

Also, by the definition of $\widehat{Q}$ as the maximal such subpolygon, for all $<_{xy}$-motononic hitting sets $p <_{xy} q$, it holds that q lies in the interior of $\widehat{Q}$.

The aim of this section is to prove the following lemma, which will be used later in Section 5.4 as a subroutine, when extending the scenario to arbitrary-length sequences.

Lemma 5.7. *The maximal subpolygon $\widehat{Q}$ of Q with no $<_{xy}$-infeasible points can be computed in $O(|P| + |Q|)$ time.*

This additionally yields the following lemma.


(a) Q' entirely below the left-lower hull of P.

(b) Q' entirely above the left-lower hull of P.

(c) Q' and the left-lower hull of P intersect.

Figure 5.2: The three possible cases how Q' can interact with the left-lower-hull of P.

Lemma 5.8. *Computing a $<_{xy}$-monotonic hitting set $p <_{xy} q$ for $P <_{xy} Q$ takes $O(|P| + |Q|)$ time.*

We now discuss $<_{xy}$-infeasible points in greater detail. When obvious from the context, we will omit the qualifier *regarding P* for easier reading. Let $v = (x_v, y_v)$ be the leftmost vertex on the left-lower hull of P. Any point $q = (x_q, y_q)$ of Q with $x_q < x_v$ is trivially $<_{xy}$-infeasible as illustrated in Figure 5.1(b) by the brown vertical dashed line. Analogously any point below the lowest vertex $w = (x_w, y_w)$ of the left-lower hull of P is $<_{xy}$-infeasible as shown by the brown horizontal dashed line in Figure 5.1(b). To cut off these trivially $<_{xy}$-infeasible points we intersect Q with the two halfplanes (oriented right and up) corresponding to the vertical line through v and the horizontal line through w. We call the polygon Q without these two trivially $<_{xy}$-infeasible regions Q'.

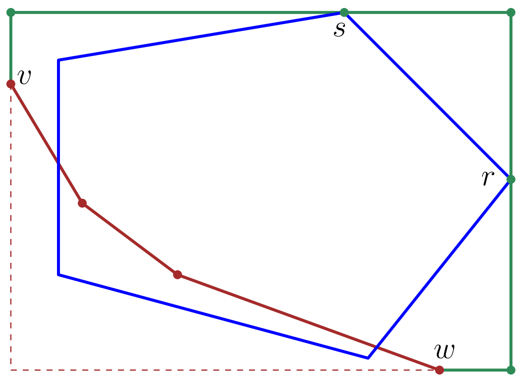
If Q' is empty there is no $<_{xy}$-feasible point and thus no $<_{xy}$-monotonic hitting set. Any point q with $x_q > x_w$ is trivially $<_{xy}$-feasible as $y_q > y_w$ for all points of Q'.

Let $\varphi : [x_v, x_w] \to [y_w, y_v]$ be the function that maps an x-coordinate to its corresponding y-coordinate on the left-lower hull of P. The left-lower hull of P is strictly monotonically decreasing, therefore φ is bijective. Points with x_q in the interval (x_v, x_w) are $<_{xy}$-feasible if and only if $y_q > \varphi(x_q)$.

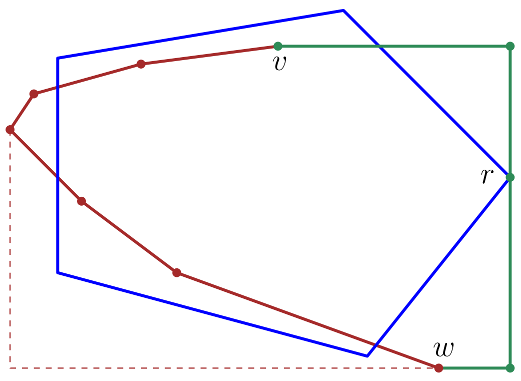
By contraposition the only remaining $<_{xy}$-infeasible points of Q' are those *below* the left-lower hull of P. If non-empty, there are three possible cases how Q' can interact with the left-lower hull of P. They are illustrated in Figure 5.2.

- Q' lies entirely below the left-lower hull of P. In this case Q' is completely infeasible and there is no $<_{xy}$-monotonic hitting set for $P <_{xy} Q$.
- Q' lies entirely above the left-lower hull of P. In this case Q' is completely feasible.
- Q' intersects the left-lower hull of P. In this case the regions below the left-lower hull of P are infeasible and those above are feasible.

As the first two cases are trivial, in the following we focus on the algorithmic details of the third case. By cutting off the remaining $<_{xy}$-infeasible regions of Q', which lie between the horizontal through w, the vertical through v and the left-lower hull of P, we finally obtain $\widehat{Q}$. Recall that $\widehat{Q}$ is the maximal subpolygon of Q, where all points in interior($\widehat{Q}$) are $<_{xy}$-feasible. In the following we describe how to compute $\widehat{Q}$ from Q' using a linear-time convex polygon intersection algorithm.



(a) Q' in blue. $W_{<_{xy}}$ in brown (left-lower hull of P) and green (top and right side of the rectangular bounding box of Q' plus extensions). $\widehat{Q} = Q' \cap W_{<_{xy}}$.



(b) Q' in blue. $W_{<_x}$ in brown (left hull of P) and green (right side of the rectangular bounding box of Q' plus extensions). $\widehat{Q} = Q' \cap W_{<_x}$.

Figure 5.3

Convex Polygon Intersection

We can obtain $\widehat{Q}$ as an intersection polygon, using a linear-time convex polygon intersection algorithm such as by Toussaint[Tou85] or O'Rourke et al.[OCON82]. Let $r = (x_r, y_r)$ be the rightmost point of the right-upper hull of Q', $s = (x_s, y_s)$ be the topmost point of the right-upper hull of Q' as illustrated in Figure 5.3(a). It is easy to see that $\widehat{Q}$ is the intersection polygon of Q' and the polygon $W_{<_{xy}}$ consisting of the left-lower hull of P as well as, in counter-clockwise order, $\langle w, (x_r, y_w), (x_r, y_s), (x_v, y_s), v\rangle$. The idea is to use the top and right side of the rectangular bounding box of Q', illustrated in green. The bottom and left sides stem from the left-lower hull of P, illustrated in brown. These are extended towards the top and right sides, with the extension in green again.

Running Time

As the running time of the convex polygon intersection algorithm is linear and the representation is explicit, we can compute $\widehat{Q}$ in $O(|P| + |Q|)$ time. An argument, why $\widehat{Q}$ has size linear in $|Q|$, is the fact that any edge of Q can cross at most two edges of P, which therefore yields $|\widehat{Q}| \leq 3|Q|$. This concludes the proof of Lemma 5.7.

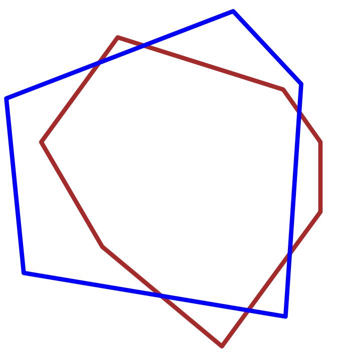
5.3 Infeasible Regions for $P <_x Q$ and $P <_y Q$

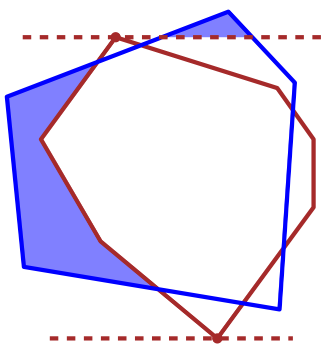
Definition 5.9 ($<_x$-infeasibility regarding P)**.** *We call a point $q = (x_q, y_q)$ in the interior of Q* $<_x$-infeasible *regarding P, if there is no point $p = (x_p, y_p)$ in the interior of P with $x_p < x_q$ and $y_p = y_q$.*

Obtaining the $<_x$-feasible remainder $\widehat{Q}$ of Q regarding P works similarly to $<_{xy}$. We now highlight the small differences.

In addition to cutting off to the left of the vertical line through P's leftmost vertex and below the horizontal line through P's lowest vertex, we also have to cut off Q above the horizontal line through P's topmost vertex. Figure 5.4(b) illustrates the horizontal halfplane cuts in dashed brown.

We furthermore need to consider the left-upper hull of P and cut off anything above it, in addition to the left-lower hull. By combination we cut off anything to the left of the left hull, visualized as the blue infeasible region in Figure 5.4(b).


(a) Two polygons P in brown and Q in blue.


(b) Infeasible regions of Q regarding $P <_x$.

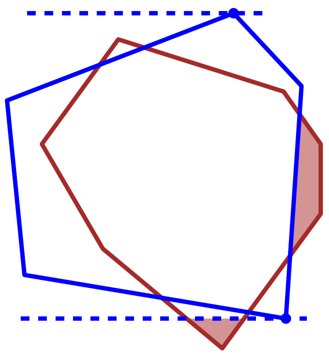

(c) Infeasible regions of P regarding $<_x Q$.

Figure 5.4

It is possible to perform the cuts in two separate steps, one for left-lower one for left-upper hull, without affecting the running time. For the sake of completeness, in Figure 5.3(b) we show the adapted polygon (brown and green) for performing the cuts in one step. We only take the right side of the rectangular bounding box of Q'. Then we extend the left hull of P by horizontal lines to the right side of the bounding box of Q'. Denote by v the topmost vertex of P's left hull, by w the lowest vertex and by r the rightmost vertex of Q'. Analogously to Section 5.2, the polygon $W_{<_x}$ to intersect Q' with, consists of the left hull of P and in counter-clockwise order $\langle w, (x_r, y_w), (x_r, y_v), v \rangle$. As in the previous section, we obtain the following lemma, which is essential for handling arbitrary-length sequences.

Lemma 5.10. *For every point $q \in \mathrm{interior}(\widehat{Q})$ there is a point $p \in \mathrm{interior}(Q)$ such that $p <_x q$ is a $<_x$-monotonic hitting set of $P <_x Q$.*

Using the algorithm modifications described above we obtain.

Lemma 5.11. *The maximal subpolygon $\widehat{Q}$ of Q with no $<_x$-infeasible points can be computed in $O(|P| + |Q|)$ time.*

$P <_y Q$ is analogous to $<_x$ by rotating the problem.

5.4 Extending to a Sequence

In this section we discuss how to extend the previous result for two polygons $P_1 \lhd P_2$ to a sequence of polygons $\mathcal{P} = \langle P_1, P_2, \ldots, P_m \rangle$ with comparisons $\mathcal{S} = \langle \lhd_1, \ldots, \lhd_{m-1} \rangle$.

We compute a sequence of polygons $\langle P_1, \widehat{P}_2, \ldots, \widehat{P}_m \rangle$, where each polygon is feasible regarding the prefix of polygons up to itself. Due to the monotonicity of $\{<_{xy}, <_x, <_y\}$, we can obtain the polygon $\widehat{P}_m$ with $\mathrm{interior}(\widehat{P}_m) \subset \mathrm{interior}(P_m)$ of all feasible points in P_m by a simple left-to-right traversal. We first cut off the $\lhd_1$-infeasible regions of P_2 regarding P_1 using the appropriate algorithm from either Section 5.2 or Section 5.3 to obtain $\widehat{P}_2$. We repeat this step with $\widehat{P}_i$ and P_{i+1} for increasing $i = 2, \ldots, m-1$ to obtain $\widehat{P}_{i+1}$ and ultimately $\widehat{P}_m$.

Theorem 5.12. *$\mathcal{P}$ has a $\mathcal{S}$-monotonic hitting set, if and only if $\langle P_1, \widehat{P}_2, \widehat{P}_3, \ldots, \widehat{P}_m \rangle$ does not contain an empty polygon.*

Proof. Assume $\langle P_1, \widehat{P}_2, \widehat{P}_3, \ldots, \widehat{P}_m \rangle$ does not contain an empty polygon. $\widehat{P}_m$ only contains points that are $\lhd_{m-1}$-feasible regarding $\widehat{P}_{m-1}$. Choose some p_m in $\mathrm{interior}(\widehat{P}_m)$.

By Lemma 5.6 or Lemma 5.10 (whether $\lhd_{m-1} =<_{xy}$ or $\lhd_{m-1} \in \{<_x, <_y\}$) there is a point p_{m-1} in interior($\widehat{P}_{m-1}$) such that $p_{m-1} \lhd_{m-1} p_m$. Since $\widehat{P}_{m-1}$ only contains $\lhd_{m-2}$-feasible points regarding $\widehat{P}_{m-2}$, by induction there is a $\langle \lhd_1, \ldots, \lhd_{m-2} \rangle$-monotonic hitting set $\langle p_1, \ldots, p_{m-1} \rangle$ of $\langle P_1, \ldots, P_{m-1} \rangle$. Due to the monotonicity of $<_{xy}, <_x, <_y$, $\langle p_1, \ldots, p_{m-1}, p_m \rangle$ is a $\mathcal{S}$-monotonic hitting set of $\mathcal{P}$.

Now assume that $\mathcal{P}$ has a $\mathcal{S}$-monotonic hitting set $p_1, \ldots, p_m$. Then the point p_i prevents p_{i+1} from being removed during the computation of $\widehat{P}_{i+1}$. Therefore $\widehat{P}_{i+1} \neq \emptyset$. $\square$

Corollary 5.13. *A $\mathcal{S}$-monotonic hitting set of $\mathcal{P}$ can be computed in $O(\sum_{i=1}^{m} |P_i|)$. In the same time we can obtain a witness (empty feasible polygon) if no $\mathcal{S}$-monotonic hitting set exists.*

Using Lemma 5.7 and Lemma 5.11 for the running time of the individual removal steps, Corollary 5.13 follows immediately.

5.5 Use in Straight-Line Aligned Level Drawing with Two Levels

In this section we reduce Straight-Line Aligned Level Drawing in graphs with two levels to solving a Monotonic Polygon Hitting Set instance. Let $G = (V, E, lvl, \prec)$ be a plane level graph with $k := |lvl(V)| = 2$, let $\mathcal{A} = \{\mathcal{R}_1, \ldots, \mathcal{R}_z\}$ be a pseudoline arrangement with respect to G such that $\mathcal{A} + \text{lvl}(V)$ is stretchable and let $A + \text{lvl}(V)$ be a stretching of $\mathcal{A} + \text{lvl}(V)$. For conciseness, we will assume that every edge has a simple canal polygon (i.e. does not cross pseudoline intersections) but justify beforehand that incorporating edges with non-simple canal polygons is possible. Recall from Section 4.4.2 that the straight-line intersection-preserving drawings of edges with non-simple canal polygons are represented by either a line segment (if trajectory(e) crosses one pseudoline intersection) or at most one single point in $\mathbb{R}^2$ (two or more pseudoline intersections). These are interpreted as extreme cases of polygons with an empty interior and the definition of infeasibility regarding these extreme polygons must be adapted to include points on their boundary, i.e. the line segment or point.

Now assume that every edge has a simple canal polygon. The set E of edges is ordered from left to right by the combinatorial embedding $\prec$ and let $e_1, \ldots, e_m$ be that ordering. Let $e = uv \in E$ be an edge with $lvl(u) = 1, lvl(v) = 2$. Recall from the definition of EDP(e), see Section 4.4.2, that each point (x_u, x_v) in the interior of EDP(e) represents an intersection-preserving drawing of e with x_u as the x-coordinate of u and x_v as the x-coordinate of v. Let $\mathcal{P} = \langle \text{EDP}(e_1), \ldots, \text{EDP}(e_m) \rangle$ be the sequence of edge drawing polygons in left-to-right ordering of the edges induced by $\prec$. The comparison operator $\lhd_i$ for two edge drawing polygons EDP(e_i) $\lhd_i$ EDP(e_{i+1}) is

- $<_{xy}$ if e_i and e_{i+1} are independent
- $<_x$ if e_i and e_{i+1} are incident in a vertex on level 1
- $<_y$ if e_i and e_{i+1} are incident in a vertex on level 2

and the comparison operator sequence $\mathcal{S} = \langle \lhd_1, \ldots, \lhd_{m-1} \rangle$ consists of the comparison operators between two consecutive polygons. By Lemma 4.14 every edge drawing polygon is convex. This yields the following lemma.

Lemma 5.14. *$\mathcal{P}$ has a $\mathcal{S}$-monotonic hitting set, if and only if (G, A) has a straight-line aligned level drawing.*

With this we obtain the following running time to compute straight-line aligned level drawings, if $k = 2$.

Lemma 5.15. *There is a* $O(\sum_{e \in E} \zeta(e)) \subset O(mz)$ *time algorithm to decide* Straight-Line Aligned Level Drawing*, if* $k = 2$.

Proof. By Lemma 4.15, computing all edge drawing polygons is done in $O(\sum_{e \in E} |\operatorname{CP}(e)|)$ time and by Corollary 5.13 solving the according Monotonic Polygon Hitting Set instance (Lemma 5.14) takes $O(\sum_{e \in E} |\operatorname{CP}(e)|)$ time. The reduction takes $O(m)$ time. From Lemma 4.12 we have $|\operatorname{CP}(e)| \in O(\operatorname{span}(e) + \zeta(e))$ and in two-level graphs we have $\operatorname{span}(e) = 1$ for every edge. The inequality $\sum_{e \in E} \zeta(e) \leq mz$ is trivial. $\square$

This is a significant improvement over the $O([|V| + \sum_{e \in E} \zeta(e) + \operatorname{span}(e)]^{3.5} + L^2)$ solution with linear constraint programming, confer Lemma 4.16. The improved algorithm is in fact asymptotically optimal, since the input must be read. The term $\Omega(\sum_{e \in E} \zeta(e) + \operatorname{span}(e)$ is a lower bound for the input size.

6. Miscellaneous Combinatorial Results

This chapter contains two miscellaneous results. In Section 6.1 we study the Intersection Sequence Embedding problem. Given a level-planar graph G without combinatorial embedding, an unrelated pseudoline arrangement and a description of how the graph is supposed to intersect the pseudoline arrangement, Intersection Sequence Embedding asks whether G has a level-planar combinatorial embedding that realizes the desired intersections. We present an efficient algorithm for Intersection Sequence Embedding in Lemma 6.2. Inspired by Mchedlidze et al. [MRR17], we additionally study the Pseudoline Existence problem. Given a plane level graph G it asks whether there is a pseudoline with respect to G through a given set $S \subset V$ of vertices (see Section 6.2).

6.1 Intersection Sequence Embedding

The Straight-Line Aligned Level Drawing problem requires a plane level graph and in particular a pseudoline arrangement with respect to the graph. In this section we spin the problem around and start with a pseudoline arrangement not with respect to a graph and a level graph without combinatorial embedding. Given a description of how the level graph shall intersect the pseudoline arrangement, we ask whether it has a combinatorial embedding which achieves the desired intersections.

Let $G = (V, E, \mathrm{lvl})$ be a level-planar graph without combinatorial embedding and let $\mathcal{A} + \mathrm{lvl}(V) = \{\mathcal{R}_1, \dots, \mathcal{R}_z\} \cup \mathrm{lvl}(V)$ be the combined arrangement of pseudolines and the levels of V. Furthermore, let $\mathrm{cells}(\mathcal{A})$ be the cells in the cell complex of $\mathbb{R}^2$ induced by $\mathcal{A}$, let pseudoline-intersections($\mathcal{A}$) be the set of pseudoline-intersections in $\mathcal{A}$ and let pseudosegments($\mathcal{A}$) be the set of pseudosegments of $\mathcal{A}$ between pseudoline-intersections. Let location : $V \to (\mathrm{cells}(\mathcal{A}) \cup \mathrm{pseudosegments}(\mathcal{A}) \cup \text{pseudoline-intersections}(\mathcal{A}))$ be a mapping of vertices to cells, pseudosegments and pseudoline intersections, indicating where that vertex is supposed to be drawn. We require that the intersection of $\mathrm{lvl}(v)$ and $\mathrm{location}(v)$ is not empty.

For some subset $E' \subset E$ and for each edge $e' = uv \in E'$ we are given a simple y-monotone curve $\alpha(e')$ through $\mathcal{A}$ that represents the combinatorial intersection structure between e' and $\mathcal{A}$. We require that the two endpoints of $\alpha(e')$ lie in $\mathrm{location}(u), \mathrm{location}(v)$. Additionally, the curves of two distinct edges do not intersect.

For each pseudoline $\mathcal{R}$ of $\mathcal{A}$, the curves $\alpha(E')$, levels and vertices of G as well as other pseudolines induce a sequence $\mathrm{iseq}(\mathcal{R})$ of edges, levels, vertices and other pseudolines that $\mathcal{R}$ is supposed to intersect along its trajectory. Note that here we do not start with an intersection sequence of objects that $\mathcal{R}$ does intersect. Instead we start with a sequence that $\mathcal{R}$ is supposed to intersect and we ask whether there is a combinatorial embedding of G which realizes this intersection sequence.

Definition 6.1 (Intersection Sequence Embedding)**.** *The* Intersection Sequence Embedding *problem asks whether G has a level-planar combinatorial embedding $\prec$ such that for each pseudoline $\mathcal{R} \in \mathcal{A}$ the sequence of intersected edges, levels, vertices and pseudolines is* $\operatorname{iseq}(\mathcal{R})$*. If yes, $\prec$ is called an* intersection sequence embedding *of* $(G, \mathcal{A}, \operatorname{iseq}(\mathcal{A}))$*.*

Although, counterintuitive at first glance, this problem is not trivial as the difficulty lies in embedding the free edges. Also note that the vertices induced by E' do not induce a partial embedding.

Lemma 6.2. Intersection Sequence Embedding *can be solved in*

$$O\left(\sum_{i=1}^{l} |\operatorname{iseq}(\mathcal{R}_i)| + |V|\right)$$

time.

For the proof we need some tools. Instead of considering the intersection sequence along each pseudoline we will consider the intersection sequence around the boundary of each cell $c \in \operatorname{cells}(\mathcal{A})$. Define cyclic-iseq$(c)$ as the cyclic, ordered sequence of curves $\alpha(E')$, levels, vertices and pseudolines intersected by the pseudolines bounding c. We also call cyclic-iseq(c) the *cyclic-intersection-sequence* of c.

Remark 6.3. *The boundary of each cell c can be split into a left and a right hull, where the left hull has the interior of c to the right and the right hull has the interior of c to the left. In particular, if c is bounded, one pseudoline intersection is distinguished as the lowest point on the boundary, as well as another as the highest point. Unbounded cells are artificially closed with curves that intersect no edge and contain no vertex.*

The *cyclic-intersection-sequence-induced* subgraph $G[c]$ is the subgraph of G induced by the vertices in the interior of c and on the boundary of c. Additionally, the edges in cyclic-iseq(c) are added. Then they are cut off at their intersection with the boundary of c, where a dummy vertex is inserted on a dummy level on the appropriate pseudosegment of the boundary. The dummy levels due to dummy vertices on the left, respectively right hull are sorted according to the order on the left, respectively right hull and inserted between the according levels of G. For an ordering of the dummy levels from both hulls we consider edges between dummy vertices of the two hulls. Let $e = uv$ be an edge in cyclic-iseq(e) with u and v not in the interior of c, and $\operatorname{lvl}(u) < \operatorname{lvl}(v)$; see for example the edge u_1u_10 in Figure 6.1. Then the dummy vertices u' and v' due to u and v have dummy levels, with the condition $\operatorname{lvl}(u') \leq \operatorname{lvl}(v')$. Equality only holds for dummy vertices on the highest or lowest polygon-vertex on the boundary of c, i.e. when $u' = v'$.

In Figure 6.1 the dummy vertices are represented as white squares. The figure also shows an edge between the left and right hull which forces the dummy level due to the left hull vertex below the dummy level due to the right hull vertex.

Observe that the notion of cyclic-intersection-sequence-induced subgraphs is similar but not equivalent to cell-induced subgraphs used in Section 4.3 and Theorem 4.7. The cell-induced subgraph contains vertices for line intersections whereas a cyclic-intersection-sequence-induced subgraph does not. Furthermore, cell-induced subgraphs assume a given line-arrangement and embedded graph, whereas cyclic-intersection-sequence-induced subgraphs only assume an order in which edges intersects its boundary. Due to their similarity we write both as $G[c]$ but mention their differences to avoid confusion.

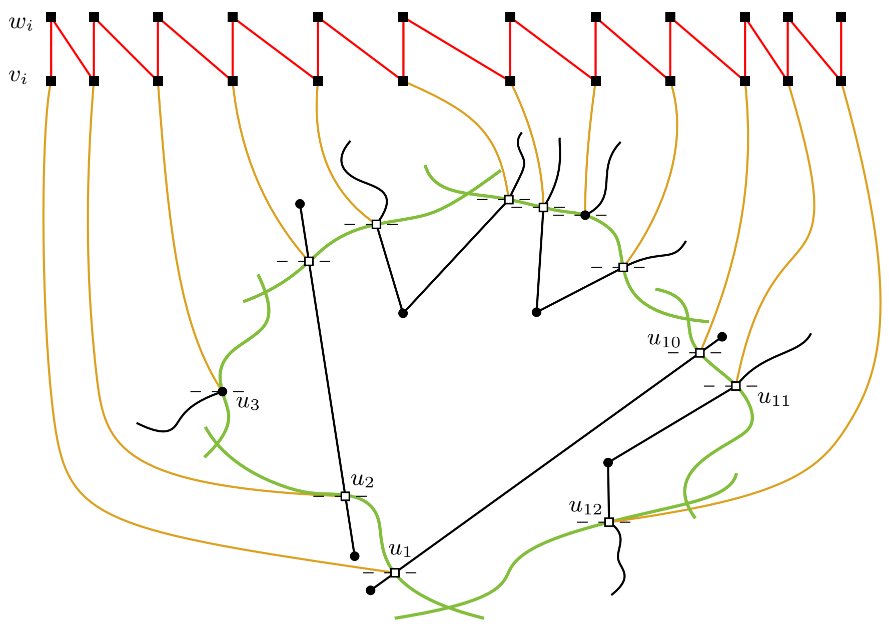


Figure 6.1: Reduction to level-planarity from the proof of Lemma 6.2 .

Lemma 6.4. *There is an intersection sequence embedding of* $(G, \mathcal{A}, \text{iseq}(\mathcal{A}))$ *if and only if for each cell* $c \in \text{cells}(\mathcal{A})$ *there is a level-planar embedding* $\prec_c$ *of* $G[c]$ *such that the boundary of* c *intersects and visits vertices, dummy vertices due to edges, levels and other pseudolines in the order* cyclic-iseq(c).

Proof. The pseudolines $\mathcal{R}_1, \ldots, \mathcal{R}_z$ intersect E' according to iseq$(\mathcal{A})$ if and only if for each cell c the cell boundary intersects E' according to cyclic-iseq(c).

Given an intersection sequence embedding $\prec$ of $(G, \mathcal{A}, \text{iseq}(\mathcal{A}))$ we obtain an embedding of $G[c]$ by taking the embedding of the vertices and edges in c as well as vertices on the boundary of c and edges intersecting the boundary. Each dummy vertex due to an intersected edge is assigned to a level. It is placed where its intersected edge crosses that level in $\prec$.

Conversely given a combinatorial embedding $\prec_c$ of $G[c]$ for each cell $c \in \text{cells}(\mathcal{A})$, we obtain an embedding of G by deleting dummy vertices and dummy levels as well as reversing the subdivision of intersected edges, i.e. route intersected edges through their former dummy vertices. □

Definition 6.5 (Flipped Embedding)**.** *Let* $G = (V, E, \text{lvl}, \prec)$ *be a plane level graph. The flipped embedding is defined as the embedding obtained by reversing the sequence of level-crossings and vertices on each level.*

Lemma 6.6. *Let* $G = (V, E, \prec)$ *be a plane level graph. Then* G *is also plane with the flipped embedding of* $\prec$.

The proof of Lemma 6.6 is trivial. With the now obtained toolset, we are ready to prove Lemma 6.2.

Proof of Lemma 6.2. By Lemma 6.4 it suffices to consider each cell c and cyclic-iseq(c) and it is equivalent to enforce cyclic-iseq(c) around c. We construct a new graph $G'[c]$ consisting of the cyclic-intersection-sequence-induced subgraph $G[c]$ and the following augmentations.

Add two new levels $k+1, k+2$ above c. Let cyclic-iseq(c) be oriented clockwise. Starting with the first object on the left hull of c, recall Remark 6.3, we traverse the boundary of c and cyclic-iseq(c) in clockwise order. For each vertex u_i (dummy or original) on the boundary, introduce one vertex v_i on level $k+1$ and one w_i on level $k+2$. See the black squares in Figure 6.1. Then connect $v_i w_i$ by an edge, see the red vertical edges and connect $w_i v_{i+1}$ by an edge, see the red edges with negative slope. Finally connect $u_i v_i$ by an edge, see the golden curvy edges.

In a level-planar embedding of $G'[c]$ the clockwise order of the u_i around the boundary of c is the same as the order of the v_i on level $k+1$, otherwise two of the golden edges would cross.

The subgraph induced by all v_i and w_i (black squares, red edges) is edge-maximal level-planar and has a unique combinatorial embedding, up to a flip (recall Definition 6.5), namely $v_1, v_2, v_3, \ldots$ plus $w_1, w_2, w_3, \ldots$ and the flipped embedding. By Lemma 6.6 we can assume it is $v_1, v_2, \ldots$.

Thereby Intersection Sequence Embedding reduces to testing level planarity of $G'[c]$.

The construction of all $G'[c]$ requires

$$O\left(\sum_{i=1}^{z} |\operatorname{iseq}(\mathcal{R}_i)| + |V|\right)$$

time, as each pseudosegment induced by pseudoline-intersections and levels bounds two cells and each edge intersection introduces one dummy vertex.

Using the according linear-time embedding algorithm by Jünger et al. [JL99], we can obtain a combinatorial embedding for $G[c]$ for each c and patch them together using Lemma 6.4. □

6.2 Pseudoline Existence

Let $G = (V, E, \mathrm{lvl}, \prec)$ be a plane level graph and let $S \subset V$ be a subset of vertices. In this section we investigate the question whether there is a pseudoline with respect to G that collects all vertices of S. This is called the Pseudoline Existence Problem. In the spirit of Mchedlidze et al. [MRR17] we want to find a characterization of when such a pseudoline exists. For general planar graphs with fixed topological embedding, Theorem 4.9 due to DaLozzo et al. [DLDF$^+$16], states that there is a straight-line drawing in which the vertice of S are collinear if and only if there is a pseudoline through S. We proved the equivalent for proper plane level graphs in Chapter 4, see Corollary 4.4. Mchedlidze et al. [MRR17] give a polynomial time transformation of Pseudoline Existence in general planar graphs to the K-Cycle problem, which is the decision problem that asks whether a graph has a simple cycle through a set of given vertices. By a result due to Wahlström [Wah13], this shows Pseudoline Existence is fixed-parameter tractable in general planar graphs.

In this section we present a similar result, where we reduce Pseudoline Existence to the question whether another graph has a simple path with some special properties, see Theorem 6.10. For plane proper level graphs we then show in Theorem 6.11 that the necessary condition "$G[S]$ is a level-ordered linear forest" (confer Definition 6.8 and Remark 6.9) is also sufficient for pseudoline existence.

Definition 6.7. *Let $G = (V, E, \mathrm{lvl})$ be a level graph. A set $S \subset V$ of vertices is called y-monotonic, if each level contains at most one vertex of S.*

Definition 6.8. *A* level-ordered linear forest *on a y-monotonic set $S \subset V$ is a level graph $H = (S, E_H)$ consisting of independent paths, such that no triple of vertices $v_1, v_2, v_3 \in S$ exists with* $\text{lvl}(v_1) < \text{lvl}(v_2) < \text{lvl}(v_3)$ *and* $(v_1, v_3) \in E_H$*.*

More colloquially, an upward edge from some vertex v can only connect it to the vertex w with the next-higher level, i.e. cannot skip a vertex between v and w.

Remark 6.9. *Let $S \subset V$ be a y-monotonic set of vertices. If $G[S]$ is not a level-ordered linear forest, there is no pseudoline through S with respect to G.*

This states a necessary condition for the vertices of S. Therefore we assume $G[S]$ to be a level-ordered linear forest for the following construction. In the following we describe the steps to obtain, what is called the *extended dual* G^*_e of G. The procedure is very similar to that in [MRR17]. We use it to prove Theorem 6.10. Its goal is to illustrate a correspondence between pseudolines through S and a level-monotonic *st*-path from the lowest to the highest level in the extended dual.

Step 1: Subdivide each edge e with dummy vertices at its level-crossings to make G a plane proper level graph. Denote by rev-subdivision(e') the original edge e which has e' as one of its subdivided parts.

Let G' be the infinite track-plane extension of G (confer Definition 2.7). Then its dual G'^* is a level st-plane track graph with $k+1$ levels, see Figure 6.2(a), with source s on the bottom level and sink t on the top level of G'^*. The inter-level edges of G'^* are oriented upward.

Step 2: Place each vertex v of S into its corresponding face in G'^*.

We construct the extended dual G^*_e by combining G'^* and S in one graph. Therefore we mix the levels and assign half-integer levels to the vertices of G'^* and the original integer levels in G to vertices of S. The half-integer levels are also called dual levels, whereas the integer levels are called primal levels. We assign s to level $\text{lvl}_{G^*_e}(s) := 0.5$ and t to level $\text{lvl}_{G^*_e}(t) := k + 0.5$. Let f^* be a vertex of G'^* other than s, t, that is dual to face f of G' with f enclosed between levels i and $i+1$. Then assign level $\text{lvl}_{G^*_e}(f^*) := i + 0.5$ to f^*. Vertices v of S are assigned to their original integer levels $\text{lvl}_{G^*_e}(v) := lvl(v)$.

Step 3: Let $(v_1, \ldots, v_r)$ be a directed path induced by S in G. Connect v_1 to those vertices f^* of G'^* with $\text{lvl}_{G^*_e}(f^*) = \text{lvl}_{G^*_e}(v_1) - 0.5$, whose primal face f is incident to v_1 in G'. Analogously connect v_r to vertices f^* with $\text{lvl}_{G^*_e}(f^*) = \text{lvl}_{G^*_e}(v_r) + 0.5$ whose primal face is incident to v_r (red dashed edges in Figure 6.2(b)).

Step 4: For each $v \in S$ remove the edges that are dual to some primal edge $e' \in E'$ where rev-subdivision(e') is incident to v in G. Illustrated in Figure 6.2(c).

Step 5: In each dual level $i - 0.5$ that is adjacent to one primal level i (i.e. contains a vertex of S), the upward oriented inter-level edges to the dual level $i + 0.5$ are deleted. See the gray edges deleted in Figure 6.2(d) between dual levels that enclose a red vertex of S. Furthermore, every dual level $i + 0.5$, that is enclosed between two primal levels i and $i+1$ with an edge of S between the primal levels, is deleted entirely. These two modifications force a *monotonic* path, that *visits every dual level*, to collect all vertices of S, since it is only possible to connect two consecutive dual levels, that have vertices of S between them, by visiting the according path induced in $G[S]$ (see Figure 6.2(d)).

Step 6: Let $e \in E$ be an edge of G. Let subdivision$^*(e)$ be the edges of G'^* that are dual to the subdivision of e. Then subdivision$^*(e)$ is replaced by a star with the dual edges' vertices as leaves and a single center vertex. This forces a *simple* path to use at most one of the dual edges that stem from the subdivision of e. The center vertex is not assigned to a level. Note that the expansion only affects intra-level edges of G'^* as they are dual to inter-level edges of G'. Denote the center vertex for edge e by $\mathrm{cent}(e)$ and the set of all star-centers by $\mathrm{cent}(E)$.

Theorem 6.10. *Let $G = (V, E, \mathrm{lvl}, \prec)$ be a plane level graph, $S \subset V$ a y-monotonic subset of vertices and $G[S]$ a level-ordered linear forest. There exists a pseudoline $\mathcal{R}$ with respect to G, that passes through all vertices of S if and only if $G^*_e = (V^*, E^*)$ has a simple st-path P that is monotonic in the levels of $V^* \setminus \mathrm{cent}(E)$ and visits each odd level of G^*_e.*

Proof. Assume there is a pseudoline $\mathcal{R}$ with respect to G through exactly the vertices of S. Without loss of generality assume that $\mathcal{R}$ does not visit level-crossings of edges. Then let $\mathcal{R}'$ be the pseudoline obtained by considering $\mathcal{R}$ through G'. We decompose $\mathcal{R}'$ into pieces of four different categories. For each category, we describe how to translate it into an edge or short path in G^*_e. Chained together, the pieces form the path P.

A piece of type α connecting two adjacent faces f_1, f_2 of G' by crossing an intra-level edge e of G'. Then the piece is substituted by the upward-oriented inter-level edge $e^* = (f^*_1, f^*_2)$ of G^*_e dual to e. Since the piece does not visit a vertex of S, the edge e^* is not deleted in Step 5.

A piece of type β connecting two adjacent faces f_1, f_2 of G' by crossing an inter-level edge e' of G'. Let $e = \text{rev-subdivision}(e')$. We substitute the piece by the two-edge path $(f^*_1, \mathrm{cent}(e), f^*_2)$. The edges $(f^*_1, \mathrm{cent}(e)$ and $(\mathrm{cent}(e, f^*_2)$ were not removed in Step 4 since e is not incident to some vertex of S since $\mathcal{R}$ visits these and only these vertices. Additionally, we know that β does not cross any of the G'-edges from subdivision(e) other than e'. Furthermore, no other piece of $\mathcal{R}'$ crosses any of these edges. Therefore the star-center $\mathrm{cent}(e)$ of the subdivided edges (see Step 6) is only used by this and no other pieces of $\mathcal{R}'$. Together with the upward orientation of the other following edge types, this yields that P is simple. Also note that $\mathrm{lvl}_{G^*_e}(f^*_1) = \mathrm{lvl}_{G^*_e}(f^*_2)$.

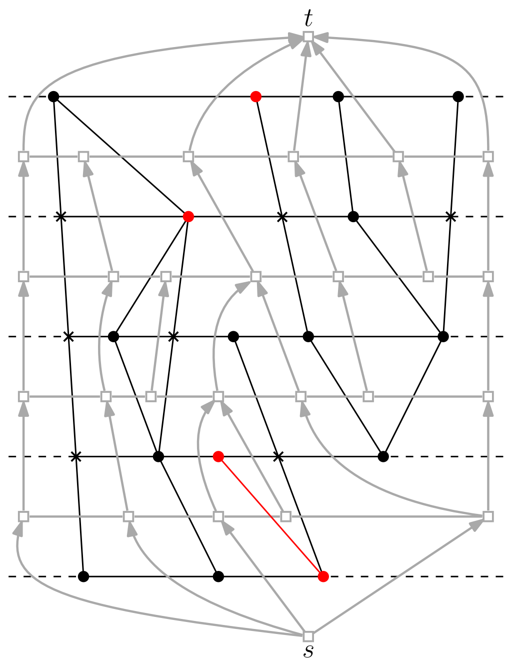
A piece of type γ connecting a face f of G' with an incident vertex $v \in S$. We know that v is the end-vertex of a path induced by S. Substitute the piece by the edge between f^* and v, that was added in Step 3.

A piece of type δ connecting two adjacent vertices v_1, v_2 of S. Substitute the piece by the edge (v_1, v_2) that was introduced in Step 2.

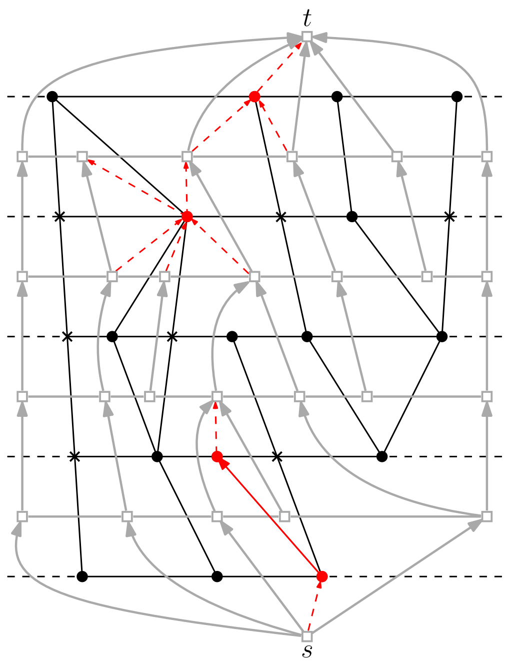
As mentioned already, P is simple. Due to the upward orientation of all edges of G^*_e except the expanded intra-level edges (Step 6), P is monotonic in the levels of $V^* \setminus \mathrm{cent}(E)$. P is an *st*-path, since s is visited first and t is visited last by $\mathcal{R}$. Furthermore, since all pieces of type β are substituted by two edges with endpoints on the same level, the star-centers are not used to skip levels. Combining this with the fact that each inter-level edge between odd levels only spans these two levels, yields that each odd level of G^*_e is visited by P.

Now assume that P is a directed, simple *st*-path in G^*_e that is monotonically increasing in the levels of $V^* \setminus \mathrm{cent}(E)$ and visits each odd level once. We construct a pseudoline $\mathcal{R}$ with respect to G through S by constructing a pseudoline $\mathcal{R}'$ with respect to G'.

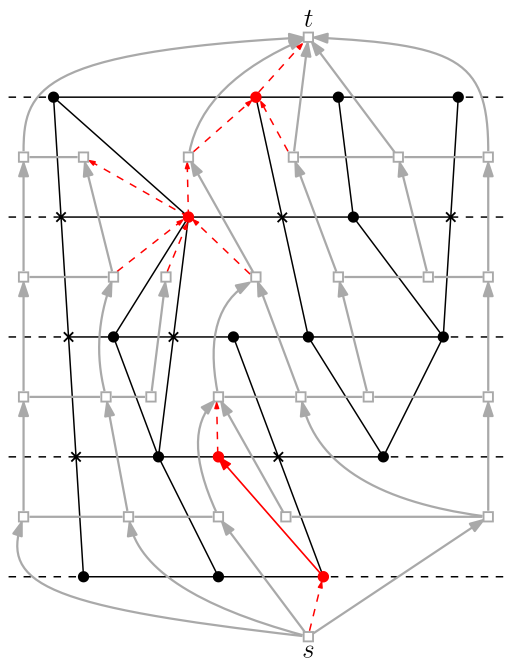
The curve $\mathcal{R}'$ is routed along $P \setminus \mathrm{cent}(E)$, i.e. the star-centers are ignored. Since each odd level has to be visited once and P is monotonic in $V^* \setminus \mathrm{cent}(E)$, each star-center $\mathrm{cent}(e)$ on P is proceeded and succeeded on P by vertices of the same odd level in G^*_e. By Step 4, no edges incident to S are crossed and any path induced by S is traversed by $\mathcal{R}'$. By Steps 2,3 and 5, all vertices of S and all edges of $G[S]$ are collected. By Step 6 each edge is crossed at most once. By the monotonicity of P, each level in G' is crossed exactly



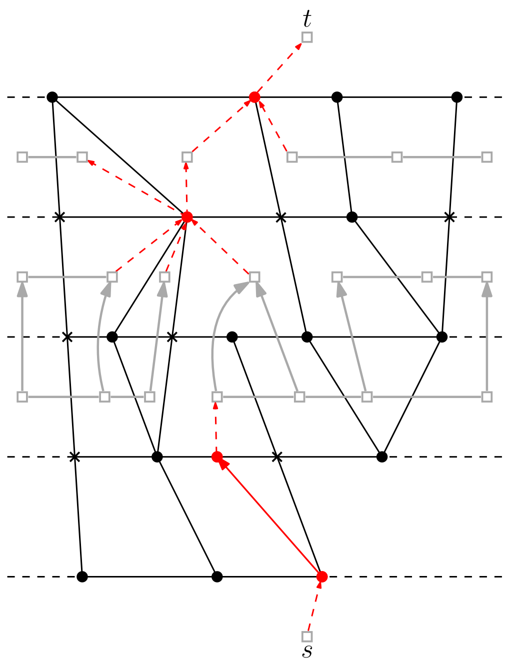
(a) G'^* in gray, G' in black. Subdivision dummy vertices drawn as crosses. The vertices and edges of S in red.



(b) After Step 3.



(c) After Step 4. The gray inter-level edges dual to edges that are incident to some vertex in S are deleted.



(d) After Step 5.

Figure 6.2: Construction of the extended dual.

once. Therefore $\mathcal{R}'$ is a pseudoline with respect to G' through the vertices of S. By deleting the augmentation that turned G into G' and the according parts on $\mathcal{R}'$ we obtain a pseudoline $\mathcal{R}$ with respect to G. □

Note that a path P as in this theorem, visits the vertices of S, simply due to the construction of G_e^* and not by making it a condition for the path. The proof is obviously similar to that of Lemma 1 in [MRR17] because it is a similar result for a different graph class. Note that in opposition to [MRR17] we need not distinguish between disconnected, 1-connected and 2-connected graphs. Additionally, the subdivision of single-edge paths induced by S is dropped since P has to be monotonic.

The extended dual G_e^* consists of directed inter-level and undirected intra-level edges where the latter are expanded to stars. The condition ”simple” for P almost suffices but does not quite. In fact the monotonicity condition and the ”each dual level in G_e^* is visited once” condition are only required to prevent level-hopping via the star-centers. We hope they can be dropped by further augmentations to enforce monotonicity by only using upward directed edges and leave such a construction as an open problem. These obstructive properties prohibit the reduction to some known fixed-parameter tractable path-finding problem (such as Ordered K-Cycle or Disjoint Paths) to solve Pseudoline Existence, in opposition to [MRR17], who reduced Pseudoline Existence in general plane embedded graphs to K-Cycle.

In proper level graphs, the star expansion is not necessary. We can even prove that the necessary condition from Remark 6.9 is sufficient for pseudoline existence.

Theorem 6.11. *Let $G = (V, E, \mathrm{lvl}, \prec)$ be a plane proper level graph and $S \subset V$ a subset of vertices. There exists a pseudoline $\mathcal{R}$ with respect to G that passes through S if and only if S is y-monotonic and $G[S]$ a level-ordered linear forest. If $G[S]$ is a level-ordered linear forest, $\mathcal{R}$ can be constructed in $O(k + |V|)$ time.*

Proof. Assume $G[S]$ is not a level-ordered linear forest. Recall that Remark 6.9 states that in this case no pseudoline with respect to G through S exists.

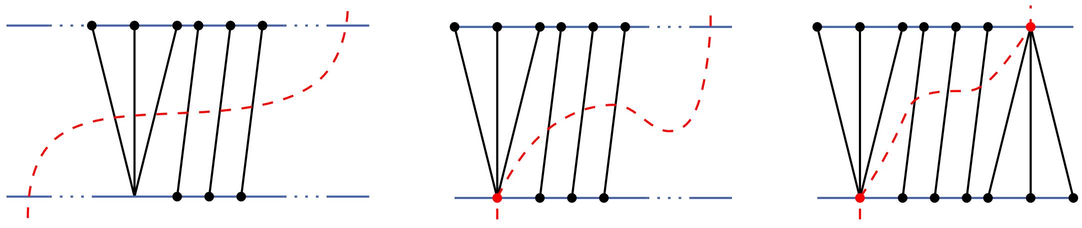
Now assume $G[S]$ is a level-ordered linear forest. We construct $\mathcal{R}$ iteratively, from level 1 to k, in the track-plane extension G' of G (confer Definition 2.6). In G' each face is either quadrangular, triangular or unbounded. Of the latter kind there are exactly two in each space between two consecutive levels, the leftmost and rightmost. Denote by space$[i]$ the space between the two consecutive levels i and $i + 1$.

The *entry face* of space$[i]$ is the first face of G' that $\mathcal{R}$ visits in space$[i]$. When $\mathcal{R}$ enters space$[i]$ between two vertices or through an unbounded segment, there is a unique entry face. Conversely, when $\mathcal{R}$ enters space$[i]$ through a vertex v of S with upward edges, we can choose the entry face among the faces of space$[i]$ that are bounded by the upward edges of v.

We now describe the iterative construction of $\mathcal{R}$ by the means of a base case and an induction step.

Induction Base: If level 1 contains a vertex $v_1 \in S$, draw a piece entering v_1, otherwise draw a piece up to level 1 which ends between two vertices or on an unbounded segment of level 1.

Induction Step $i \to i + 1$: Adding the piece of $\mathcal{R}$ that crosses space$[i]$ is done by distinguishing the possible combinations of whether S has a vertex on levels i and $i + 1$.



(a) Case 1. (b) Case 2. (c) Case 3.

Figure 6.3: Visualization of the three cases in the induction step of Theorem 6.11.

- **Case 1.** $\mathrm{lvl}^{-1}(\{i, i+1\}) \cap S = \emptyset$

  From the entry face f in space$[i]$ draw a y-monotone curve through all faces to the right of f, up to the unbounded right face, then end the curve on level $i+1$ on the right unbounded segment. This ensures that no vertex $v \notin S$ is visited by $\mathcal{R}$ (see Figure 6.3(a)).

- **Case 2.** $\mathrm{lvl}^{-1}(i+1) \cap S = \emptyset$ and $\mathrm{lvl}^{-1}(i) \cap S = \{v_i\}$

  Similar to Case 1 but choose the entry face f as the rightmost face that is bounded by the rightmost upward edge of v_i (see Figure 6.3(b)).

- **Case 3.** $\mathrm{lvl}^{-1}(i+1) \cap S = \{v_{i+1}\}$ and $\mathrm{lvl}^{-1}(i) \cap S = \{v_i\}$ and $v_i v_{i+1} \notin E[S]$

  If some face f, bounded by the upward edges of v_i is incident to v_{i+1} choose f as the entry face (and exit face). Otherwise, if $v_{i+1} \prec_{i+1} N^+(v_i)$ choose the leftmost face bounded by the leftmost upward edge of v_i as the entry face. Choose the rightmost face bounded by the rightmost upward edge if $N^+(v_i) \prec_i v_{i+1}$. Then route the drawing to the nearest face that is incident to v_{i+1}, where nearest is with respect to the dual path of faces in space$[i]$ (see Figure 6.3(c)).

- **Case 4.** $\mathrm{lvl}^{-1}(i+1) \cap S = \{v_{i+1}\}$ and $\mathrm{lvl}^{-1}(i) \cap S = \{v_i\}$ and $v_i v_{i+1} \in E[S]$

  Route the drawing along $v_i v_{i+1}$.

- **Case 5.** $\mathrm{lvl}^{-1}(i+1) \cap S = \{v_{i+1}\}$ and $\mathrm{lvl}^{-1}(i) \cap S = \emptyset$

  From the unique entry face, route the drawing to the nearest face that is incident to v_{i+1}.

$\square$

Remark 6.12. *An alternative proof to Theorem 6.11 is to use Theorem 6.10. First we note that G_e^* always contains a st-path and thereby a simple st-path. The difficulty lies in finding one that is monotonic in the levels of $V^* \setminus \mathrm{cent}(E)$ and visits all odd levels. This is trivial for plane proper level graphs G. We can omit Step 6. Then any path is monotonic because G_e^* consists only of upward and bidirected intra-level edges. Furthermore, no odd level can be skipped.*

Combining Corollary 4.4 and Theorem 6.11 we derive the following proposition.

Proposition 6.13. *A set $S \subset V$ can be drawn collinear in a plane proper level graph $G = (V, E, \mathrm{lvl}, \prec)$ if and only $G[S]$ is a level-ordered linear forest.*

7. Conclusion and Future Work

In this thesis we considered straight-line aligned level drawings and some related questions. We presented a polynomial-time algorithm for Straight-Line Aligned Level Drawing based on linear programming. It can be combined with other linear constraints for constrained graph drawing problems such as Fixed Embedding Partial Drawing Extensibility or constraining faces to be drawn as convex polygons, without additional effort. The Monotonic Polygon Hitting Set, a novel geometric problem was introduced and alongside, we proposed an asymptotically optimal algorithm to solve it, using convex polygon intersection. We then showed how to reduce Straight-Line Aligned Level Drawing for graphs with two levels to Monotonic Polygon Hitting Set and thus obtain an asymptotically optimal $O(\sum_{e \in E} \zeta(e) + \mathrm{span}(e)) \subset O(mz)$ algorithm for Straight-Line Aligned Level Drawing in graphs with two levels. This reduction relies on the convexity of edge drawing polygons, a space-efficient and efficiently computable (both $O(\zeta(e) + \mathrm{span}(e)))$ representation of the intersection-preserving drawings of a single edge e. A generalization of Monotonic Polygon Hitting Set to more than two levels, which admits a more efficient algorithm than linear programming remains an open problem.

We proved that any aligned level graph $(G, \mathcal{A})$ with stretchable $\mathcal{A}+\mathrm{lvl}(V)$ has an aligned level drawing in which every edge is at most bent on the pseudolines it intersects. Furthermore, we showed that if $\mathcal{A}$ is a parallel pseudoline arrangement or an arrangement of up to two pseudolines then for every plane proper level graph G there is a straight-line aligned level drawing of $(G, \mathcal{A})$. The Pappus configuration is an example of an aligned level graph $(G, \mathcal{A})$ (where G is proper, consists of four levels and $\mathcal{A}$ consists of six pseudolines) which does not admit a straight-line aligned level drawing. This raises the question for the smallest number of pseudolines in a proper aligned level graph which does not admit a straight-line aligned level drawing. Note that this question is related to the smallest number of pseudolines in a non-stretchable pseudoline arrangement, which is nine. From our results we obtain that this number is between two and six and leave narrowing down the answer as an open problem. Another interesting question is whether this number is smaller for non-proper graphs.

Similarly to Mchedlidze et al. [MRR17], we provide a characterization of pseudoline existence through a given set S of vertices. Due to obstructive conditions on the st-path we do not find a reduction to some path-finding problem. However, we hope that through further augmentation the obstructive conditions can be eliminated or circumvented and leave this as an open problem. For plane proper level graphs G we showed the equivalence of the existence of a pseudoline through S and the existence of a plane level drawing in which the vertices in S are collinear. Furthermore, we proved that such a pseudoline exists if and only if $G[S]$ is a level-ordered linear forest, which is an easily verifiable condition.

Finally we considered the Intersection Sequence Embedding problem, which asks whether a given intersection behavior between a level graph without combinatorial embedding and a pseudoline arrangement can be realized by finding an appropriate combinatorial embedding. We gave a linear-time reduction from Intersection Sequence Embedding to testing level planarity, which is feasible in linear time.

Bibliography

[ADF$^+$10] Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing planarity of partially embedded graphs. In Moses Charikar, editor, *Proceedings of the Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*. Society for Industrial and Applied Mathematics, 2010.

[BKM98] Therese Biedl, Michael Kaufmann, and Petra Mutzel. Drawing planar partitions II: HH-drawings. *Proceedings of the 24th Workshop on Graph-Theoretic Concepts in Computer Science (WG'98)*, 98:124–136, 1998.

[BP16] Therese Biedl and Claire Pennarun. Non-aligned drawings of planar graphs. In *Proceedings of the 24th International Symposium on Graph Drawing (GD'16)*, pages 131–143. Springer, 2016.

[BR17] Guido Brückner and Ignaz Rutter. Partial and constrained level planarity. In Philipp N. Klein, editor, *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 2000–2011. SIAM, 2017.

[Can88] John Canny. Some algebraic and geometric computations in pspace. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC'88)*, pages 460–467. ACM, 1988.

[DBT88] Giuseppe Di Battista and Roberto Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61(2-3):175–198, 1988.

[DLDF$^+$16] Giordano Da Lozzo, Vida Dujmović, Fabrizio Frati, Tamara Mchedlidze, and Vincenzo Roselli. Drawing planar graphs with many collinear vertices. In *Proceedings of the 24th International Symposium on Graph Drawing (GD'16)*. Springer, 2016.

[Duj15] Vida Dujmović. The utility of untangling. In *Proceedings of the 23rd International Symposium on Graph Drawing (GD'15)*, pages 321–332. Springer, 2015.

[EA81] Hossam El Gindy and David Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2(2):186–197, 1981.

[EFLN06] Peter Eades, Qingwen Feng, Xuemin Lin, and Hiroshi Nagamochi. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. *Algorithmica*, 44(1):1–32, 2006.

[EG85] Hossam Ahmed El Gindy. *Hierarchical Decomposition of Polygons with Applications.* PhD thesis, McGill University, Montreal, Quebec, Canada, 1985.

[FPSŠ13] Radoslav Fulek, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Hanani–Tutte, Monotone Drawings, and Level-Planarity. In *Thirty essays on geometric graph theory*, pages 263–287. Springer, 2013.

[GHL$^+$87] Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987.

[HN08] Seok-Hee Hong and Hiroshi Nagamochi. Convex drawings of graphs with non-convex boundary constraints. *Discrete Applied Mathematics*, 156(12):2368–2380, 2008.

[JL99] Michael Jünger and Sebastian Leipert. Level planar embedding in linear time. In *Proceedings of the 7th International Symposium on Graph Drawing (GD'99)*, pages 72–81. Springer, 1999.

[JLM98] Michael Jünger, Sebastian Leipert, and Petra Mutzel. Level planarity testing in linear time. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*, volume 98, pages 224–237. Springer, 1998.

[Jor93] Camille Jordan. *Cours d'analyse de l'École polytechnique*, volume 1. Gauthier-Villars et fils, 1893.

[JS87] Barry Joe and Richard B. Simpson. Corrections to Lee's visibility polygon algorithm. *BIT Numerical Mathematics*, 27(4):458–473, 1987.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC'84)*, pages 302–311. ACM, 1984.

[KR17] Boris Klemz and Günter Rote. Ordered level planarity, geodesic planarity and bi-monotonicity. *arXiv preprint arXiv:1708.07428*, 2017.

[Lee83] Der-Tsai Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22(2):207–221, 1983.

[Lev26] Friedrich Levi. Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade. *Ber. Math.-Phys. Kl. Sächs. Akad. Wiss*, 78:256–267, 1926.

[LL86] Der-Tsai Lee and Arthur K Lin. Computing the visibility polygon from an edge. *Computer vision, graphics, and image processing*, 34(1):1–19, 1986.

[Mnë88] Nikolai E. Mnëv. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In *Topology and geometry—Rohlin seminar*, pages 527–543. Springer, 1988.

[MRR17] Tamara Mchedlidze, Marcel Radermacher, and Ignaz Rutter. Aligned drawing of planar graphs. *arXiv preprint arXiv:1708.08778*, 2017.

[OCON82] Joseph O'Rourke, Chi-Bin Chien, Thomas Olson, and David Naddor. A new linear algorithm for intersecting convex polygons. *Computer Graphics and Image Processing*, 19(4):384–391, 1982.

[O'R87] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*, volume 57. Oxford University Press, 1987.

[Pat06] Maurizio Patrignani. On extending a partial straight-line drawing. *International Journal of Foundations of Computer Science*, 17(05):1061–1069, 2006.

[PT04] János Pach and Géza Tóth. Monotone drawings of planar graphs. *Journal of Graph Theory*, 46(1):39–47, 2004.

[Pur97] Helen Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, pages 248–261. Springer, 1997.

[Rad15] Marcel Radermacher. How to draw a planarization. Master's thesis, Karlsruhe Institute of Technology, 2015.

[RSB$^+$01] Bert Randerath, Ewald Speckenmeyer, Endre Boros, Peter Hammer, Alex Kogan, Kazuhisa Makino, Bruno Simeone, and Ondrej Cepek. A satisfiability formulation of problems on level graphs. *Electronic Notes in Discrete Mathematics*, 9:269–277, 2001.

[Sch09] Marcus Schaefer. Complexity of some geometric and topological problems. In *Proceedings of the 17th International Symposium on Graph Drawing (GD'09)*, pages 334–344. Springer, 2009.

[Sho91] Peter Shor. Stretchability of pseudolines is NP-hard. *Applied Geometry and Discrete Mathematics-The Victor Klee Festschrift*, 1991.

[STT81] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[TOG04] Csaba D. Toth, Joseph O'Rourke, and Jacob E. Goodman. *Handbook of discrete and computational geometry.* CRC press, 2004.

[Tou85] Godfried T. Toussaint. A simple linear algorithm for intersecting convex polygons. *The visual computer*, 1(2):118–123, 1985.

[TVW88] Robert E. Tarjan and Christopher J. Van Wyk. An $O(n \log \log n)$-time algorithm for triangulating a simple polygon. *SIAM Journal on Computing*, 17(1):143–178, 1988.

[Wah13] Magnus Wahlström. Abusing the Tutte matrix: An algebraic instance compression for the K-set-cycle problem. In *30th International Symposium on Theoretical Aspects of Computer Science, STACS'13*, pages 341–352, 2013.