Geometric Graph Drawing Algorithms
– Theory, Engineering and Experiments –

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Marcel Radermacher

aus Wipperfürth

Tag der mündlichen Prüfung:	22.01.2020
Erste Gutachterin:	Frau Prof. Dr. Dorothea Wagner
Zweiter Gutachter:	Herr Prof. Dr. Ignaz Rutter
Dritter Gutachter:	Herr Prof. Dr. Markus Chimani

In Erinnerung an Opa Bernhard.

Danksagung

Wie das Bergsteigen hat mich die Promotion wiederholt dazu aufgefordert an meine Grenzen und darüber hinaus zu gehen. Dass ich diese Grenzen kennenlernen durfte und nicht zu weit darüber hinausgegangen bin, dafür bin ich einer Reihe von Personen dankbar.

Allen voran Ignaz Rutter, der mich, trotz der großen räumlichen Distanz, stets mit neuen Herausforderungen versorgt hat und geduldig meinen wilden Vermutungen entgegnet ist. Vielen Dank für die freundlichen Einladungen nach Eindhoven und Passau, sowie den regelmäßigen telefonischen Austausch. Neben Ignaz, bin ich Thomas Bläsius, Tamara Mchedlize und meinen weiteren Co-Autoren dankbar für die vielen spannenden Fragestellungen, die schließlich in meine Dissertation eingeflossen sind. Die gemeinsame Forschung mit euch wird für mich mit viel Freude in Erinnerung bleiben. Für die Minimierung der Tippfehler und konfuser Satzbauten in meinen Aufschrieben bin ich Ignaz Rutter zum Dank verpflichtet. Torsten Ueckerdt hat letzte Unklarheiten beim Korrekturlesen einzelner Teile meiner Dissertation aufgedeckt, auch hier für vielen Dank.

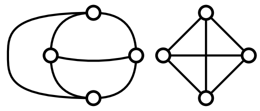
In meiner Zeit am Lehrstuhl von Dorothea Wagner durfte ich viele Länder bereisen. Die Reisen haben es mir ermöglicht viele interessante Personen und entlegene Orte kennenzulernen. Die große Freiheit die wir Doktoranden hier genießen, kann nicht ausreichend wertgeschätzt werden.

Der Austausch mit meinem Bürokollegen Lukas Barth und die Gespräche mit Tamara Mchedlidze haben mir wiederholt geholfen meine Gedanken zu sortieren. Durch die erheiternden gegenseitigen Ablenkungsmanöver von mir und Lukas wurde meine Promotion keine staubtrockene Angelegenheit. Die regelmäßigen Kaffeerunden mit meinem Kollegen haben für den notwendigen Koffeinpegel für meine Papiere gesorgt. Danke für die gute Zeit ans unserem Lehrstuhl.

Die gemeinschaftlichen Unternehmungen in den Bergen und im Ehrenamt in der Sektion Karlsruhe des Deutschen Alpenvereins haben für einen gesunden Ausgleich zwischen meiner Promotion und meiner Freizeit beigetragen.

Claudia habe ich die nötige Gelassenheit und Kraft für das letzte Jahr meiner Promotion zu verdanken. Meine Eltern Marita und Michael haben mich jederzeit bei meinen Entscheidungen unterstützt und mir den Weg freigehalten. Euer Rückhalt und euer Vertrauen haben meine Promotion erst ermöglicht.

Deutsche Zusammenfassung

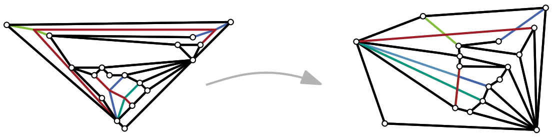


Eine planare topologische Zeichnung von K_4 und eine geometrische Zeichnung mit einer Kreuzung.

Diese Dissertation befasst sich mit theoretischen und praktischen Aspekten des Zeichnens von Graphen. Ein einfacher Graph $G = (V, E)$ kann auf vielfältige Arten dargestellt werden, abstrakt als Adjazenzmatrix oder anschaulicher als Zeichnung in der Ebene. Dabei werden die Knoten V als Punkte und die Kanten E als offene Kurven zwischen ihren Endpunkten repräsentiert. Verbietet man sich selbst schneidende Kurven, dann spricht man von einer *topologischen Zeichnung* von G. Beschränkt man die Darstellung der Kanten auf gerade Strecken, dann spricht man von einer *geometrischen* oder *geradlinigen* Zeichnung von G. Um die Qualität einer Zeichnung Γ zu bewerten, können verschiedene Qualitätsmerkmale verwendet werden, zum Beispiel die Zeichenfläche oder das Verhältnis der kürzesten zur längsten Kante. Eine besonders grundlegende Zielfunktion ist die Anzahl der Kantenkreuzungen in Γ. Imrich Vrt'o listet in seiner Online-Bibliographie [Vrt14] zu dem Thema über 700 Publikationen im Zeitraum 1954 bis 2014. Das entsprechende Optimierungsproblem ist im topologischen Fall $\mathcal{NP}$-vollständig [GJ79] und im geometrischen Fall sogar $\exists\mathbb{R}$-vollständig und somit $\mathcal{NP}$-schwer [Bie91]. Es ist bemerkenswert, dass dieses Problem, ungeachtet seiner praktischen Relevanz, ausschließlich theoretisch betrachtet wurde. Uns sind keine Implementierungen von Verfahren bekannt, die in der Lage sind für einen beliebigen Graphen eine geometrische Zeichnung mit einer kleinen Anzahl an Kreuzungen zu berechnen. Allerdings wird sogenannten kräftebasierten Verfahren die Eigenschaft zugeschrieben, für planare Graphen geometrische Zeichnungen mit einer geringen Anzahl an Kreuzungen zu berechnen [Kob13]. Ein Nachweis dieser Aussage existiert allerdings nicht.

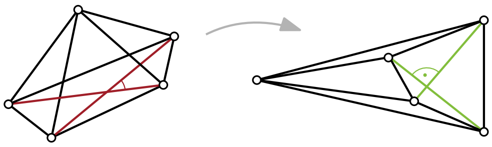
Ein wesentlicher Teil der Arbeit beschäftigt sich daher mit dem Entwurf von Verfahren zur Kreuzungsminimierung in geometrischen Zeichnungen. In einer ausführlichen experimentellen Evaluationen wird gezeigt, dass die Verfahren die Anzahl an Kreuzungen, im Vergleich zu bekannten (kräftebasierten) Algorithmen um über die Hälfte reduzieren. Da die Verfahren auf zeitaufwendigen geometrischen Operationen aufbauen, zeigen wir auf wie das Verfahren um einen Faktor 20 beschleunigt werden kann. Die entwickelten Techniken sind dabei nicht auf unsere Verfahren beschränkt, sondern in einer breiten Klasse von geometrischen Berechnungen anwendbar.

Da die minimale Anzahl an Kreuzungen in topologischen und geometrischen Zeichnungen nicht notwendigerweise übereinstimmen, kann eine zweite Perspektive auf das Problem eingenommen werden. Bei dieser wird nicht explizit die Anzahl der Kreuzungen minimiert, sondern eine topologische Zeichnung mit einer kleinen Anzahl an Kreuzungen ist vorgegeben und das Optimierungskriterium ist die Geradlinigkeit der Kanten. Die Arbeit beschreibt für dieses Problem ein kräftebasiertes Verfahren und ein Verfahren auf Grundlage von geometrischen Operationen. Die hypothesen-getriebene Evaluation zeigt, dass das zweite Verfahren signifikant bessere Zeichnungen berechnet. Die Auswertung zeigt zudem, dass die Heuristik unter bestimmten Voraussetzungen in der Lage ist geradlinige Zeichnungen zu berechnen.



Optimierung der Geradlinigkeit bei Vorgabe der Kreuzungszahl.

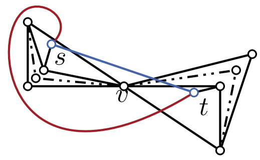
In einer verwandten Problemstellung ist nicht die Anzahl der Kreuzungen relevant, sondern der kleinste Winkel zwischen zwei sich kreuzenden Kanten in einer geometrischen Zeichnung. Gesucht ist eine Zeichnung, bei der dieser Winkel maximiert wird. Die Auswertung zeigt, dass ein randomisierter Ansatz, im Vergleich zu bekannten Verfahren, Zeichnungen mit deutlichen größeren Winkeln berechnet. In einem internationalen Wettbewerb hat diese Heuristik Zeichnungen berechnet, deren Kreuzungswinkel mindestens einen Faktor zwei größer ist als der Winkel der Kontrahenten [Dev+18].



Maximierung des kleinsten Kreuzungswinkel.

Der zweite Teil der Arbeit beschäftigt sich mit theoretischen Aspekten von geometrischen Zeichnungen *planarer Graphen*, also Graphen mit einer *kreuzungsfreien Zeichnung*. Planare Graphen haben die Eigenschaft, dass zu jeder planaren topologischen Zeichnung eine geometrische Zeichnung Γ mit den gleichen kombinatorischen Eigenschaften existiert [Fár48, Tut63]. Ein Teil der geometrischen Graphentheorie beschäftigt sich mit der Frage, ob es auch dann noch eine geometrische Zeichnung Γ von G gibt, wenn Γ zusätzliche Anforderungen erfüllen muss. Eine Modellierung einer Anforderung ist für eine Teilmenge S der Knoten die Positionen P vorzuschreiben. Entspricht die Teilmenge S der äußeren Facette eines eingebetteten planaren Graphen G und ist P in konvexer Lage, dann existiert immer eine geometrische Zeichnung von G bei der S auf P liegt [Tut63]. Für innere Facetten gilt diese Aussage nur noch unter bestimmten Voraussetzungen [MNR16]. In dieser Arbeit betrachten wir die folgende drei Anforderungen: (i) das Einbetten einer einzelnen zusätzlichen Kante mit der minimalen Anzahl an Kreuzungen, (ii) die Restriktion der Knotenpositionen auf vorgegebenen Kreisscheiben, und (iii) geometrische Zeichnungen auf Arrangements von Geraden.

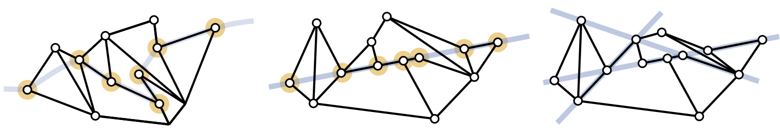
Motiviert durch die Heuristiken zur Kreuzungsminimierung untersuchen wir geometrische Zeichnungen, in denen nur eine Kante für die Kreuzungen verantwortlich ist. Formal ist eine kreuzungsfreie geometrische Zeichnung Γ von einem planaren Graphen gesucht, so dass eine gegebene neue Kante e in Γ mit der minimalen Anzahl an Kreuzungen eingefügt werden kann. Für Teilmengen der planaren Graphen wird gezeigt, dass die Zeichnung Γ effizient berechnet werden kann. Im Allgemeinen zeigen wir, dass zu dem Problem approximierende und parametrisierte Algorithmen existieren.



Die topologische Einbettung (rot) der Kante st kommt mit zwei Kreuzungen aus. Jede geometrische Einbettung (blau) hat mindestens drei Kreuzungen.

Bei den folgenden beiden Fragestellungen sind die Anforderungen durch eine interaktive Anwendung motiviert. In einem der beiden Probleme wird untersucht, ob eine geometrische Zeichnung eines planaren Graphen effizient berechnet werden kann, bei der die Positionen der Knoten auf vorgegebene Kreisscheiben eingeschränkt sind. Wir werden sehen, dass dies nur für bestimmte Mengen von Kreisscheiben der Fall und im allgemeinen $\mathcal{NP}$-schwer ist.

Je nach Anwendung werden Knoten zusätzliche Eigenschaften zugeordnet. In einem einfachen (binären) Szenario kann dies durch eine Bipartition der Knotenmenge $V = A \cup B$, mit $A \cap B = \emptyset$, formalisiert werden. In der Darstellung ist es wünschenswert, dass die Knotenmenge A und B räumlich voneinander getrennt sind. Allgemeiner wird nach einer geometrischen Zeichnung zu einem topologisch gezeichneten Graphen $G = (A \cup B \cup S, E)$ gefragt, so dass jeder Knoten in der Knotenmenge S auf einer gegebenen Geraden L platziert ist und A und B links beziehungsweise recht von L positioniert sind. Nach einer bekannten Charakterisierung existiert eine solche geometrische Zeichnung genau dann, wenn eine topologische Kurve $\mathcal{L}$ in der topologischen Zeichnung von G existiert, die die gleichen kombinatorischen Eigenschaften wie L hat [Da +18]. Wir



Die topologische Zeichnung gemeinsam mit der Kurve $\mathcal{L}$ (links) existiert genau dann, wenn eine entsprechende geometrische Zeichnung mit der Geraden L (mitte) existiert. Die Menge S ist in orange gekennzeichnet. Rechts ein Beispiel für eine geometrische Zeichnung mit einem Arrangement von Geraden.

zeigen, dass es $\mathcal{NP}$-vollständig ist zu entscheiden, ob eine solche topologische Kurve $\mathcal{L}$ existiert. Allerdings ist das Problem Fest-Parameter berechenbar in $|S|$. In einer Verallgemeinerung des Problems werden Mengen von Geraden anstelle einer einzelnen Gerade betrachtet. Wir zeigen, dass nicht zu jedem Paar von Graph und Menge von Geraden eine geometrische Zeichnung existiert. Schränkt man die Anzahl der Schnittpunkte jeder Kante mit den Geraden ein, dann folgt aus der Charakterisierung, dass jede Instanz eine geometrische Realisierung hat.

Contents

1 Introduction

Graph drawing is concerned with the automatic visualization of networks, for example, the visualization of social networks. Such a network is often modelled as a *graph* $G = (V, E)$ where a person corresponds to a vertex $v \in V$ and a friendship is an edge $uv \in E$ between two vertices u and v. One aim of graph drawing is to provide methods that layout the graph in the plane, for example, in order to help to understand and analyze the structure of the graph. There are many layout styles from which one can choose. In the most basic form, each vertex corresponds to a point in the plane and the edges that connect two vertices are represented by arbitrary curves between their endpoints; see Figure 1.1a. If the curves are non-self intersecting and have pairwise at most a single intersection point in their interior, then we refer to the drawing as a *topological drawing*. There is a lot of freedom to route edges in topological drawings, which makes it possible to represented edges by long and complicated curves. Thus, it can be difficult to track a curve from one endpoint of an edge to the other. Hence, it feels natural to restrict the *complexity* of a drawing by limiting the number of turns for each edge. In case that the edges of a drawing do not have turns, the drawing is entirely determined by the position of the vertices, i.e., edges are straight-line segments. These drawings are called *geometric drawings*; see Figure 1.1b.

This thesis studies geometric drawings from a practical and a theoretical point of view. The practical part is concerned with problems related to *crossings* in geometric drawings, i.e., an interior intersection point of edges. In the theoretical part, we study

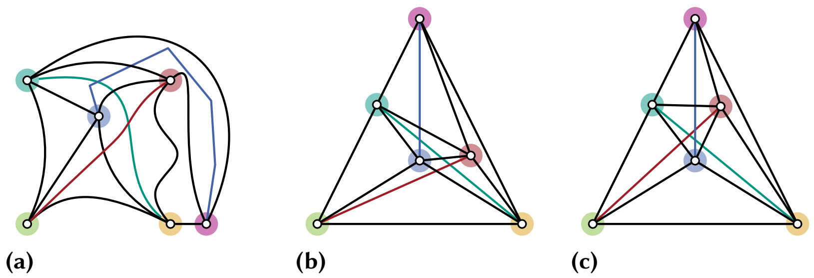


Figure 1.1: (a) A topological and (b) a geometric drawing of the same graph. The geometric drawing has the same combinatorial properties as the topological drawing in (a). Since the red edge in (c) intersects a different set of edges than the red edge in (b), (b) and (c) do not have the same combinatorial properties.

whether crossing-free drawings of *planar graphs*, i.e., graphs that have a crossing-free drawing, can be stretched to geometric drawings while satisfying prescribed constraints.

The studies of Purchase et al. [PCJ96, Pur97] indicate that the number of crossings correlates with the readability of a drawing. Thus, a fundamental quality measure of a graph drawing is the number of edge crossings, i.e., the number of edge-pairs that have an interior intersection point. Besides this practical implication of small number of crossings, the question "What is the minimum number of crossings of the complete graph K_n?" triggered the interest of many theoreticians for decades. The online-bibliography of Imrich Vrt'o [Vrt14] refers to over 700 papers on crossings in graph drawings in the time period from 1954 to 2014. The decision question whether a graph has a topological drawing with at most $k \in \mathbb{N}$ crossings, is an $\mathcal{NP}$-complete problem. If we ask for a geometric drawing, it is not known whether the problem is in $\mathcal{NP}$. Bienstock [Bie91] proved that geometric crossing minimization is $\exists\mathbb{R}$-complete, where $\exists\mathbb{R}$ is a complexity class that contains many geometric problems and for which it is unknown whether $\exists\mathbb{R} = \mathcal{NP}$. In contrast to topological drawings, there are almost no practical solution that computes geometric drawings with a minimum or at least a small number of crossings. It is claimed that so-called force-directed algorithms *tend to create crossing-free drawings for planar graphs* [Kob13]. Moreover, there are only few theoretical results that investigate the number of crossings in such drawings [CDR18]. In Part I, we will show that it is possible to compute drawings that have significantly less crossings than drawings of force-directed approaches.

The few restrictions for topological drawings make it easier, for example for a user, to construct a topological drawing of a graph. Moreover, for certain problems there already exist algorithms that compute topological drawings with high quality, for example, in case of minimizing the number of crossings. But as already observed, topological drawings can be difficult to comprehend. Thus, we can ask whether a topological drawing can be stretched to a geometric drawing while preserving the *combinatorial properties of the drawing*. Two essential combinatorial properties of a drawing are the order of the edges around a vertex and the order in which the edges cross; compare Figure 1.1b and Figure 1.1c. Unfortunately, the question whether there is a geometric drawing with the same combinatorial properties as a given topological drawing is again an $\exists\mathbb{R}$-complete problem [Mnë88, Sho91]. Thus, in general it seems to be a difficult task to find geometric drawings that preserve these properties. Fortunately, Wagner [Wag36], Fáry [Fár48], and Stein [Ste51], independently proved the following stretchability result for planar graphs.

Theorem 1.1 ([Fár48, Ste51, Wag36])**.** *For every planar topological drawing $\mathcal{E}$ of a graph G there is a planar geometric drawing of G that has the same set of edges on its boundary (outer face) as $\mathcal{E}$ and the clockwise-order of edges around each vertex in both drawings is the same.*

Thus, for every planar topological drawing there is a planar geometric drawing with the same combinatorial properties, often referred to as *a combinatorial embedding of a planar graph*. From thereon, many results on planar geometric drawings that satisfy an additional set of constraints followed. One possible constraint is to fix the positions of a subset of the vertices. More formally, let $S \subseteq V$ be a subset of the vertex set V of a planar graph $G = (V, E)$ and let P be a set of $|S|$ points in $\mathbb{R}^2$ and let $\gamma : S \to P$ be a bijective map. For a given topological drawing $\mathcal{E}$ of a planar graph G, we ask whether there is a planar geometric drawing Γ of G with the same combinatorial embedding and outer face as $\mathcal{E}$ that *extends* γ, i.e., such that for each vertex $v \in S$, v has in Γ the position $\gamma(v)$. In case that S is the set of all outer vertices and γ induces a convex drawing of the outer face of G, then there exists a geometric drawing of G that extends γ [Tut63]. If S corresponds to an inner face this is not always possible [MNR16]. Another constraint restricts the positions of the vertices in S to a straight line. Formally, we ask for straight line L and a geometric drawing Γ where all vertices in S are on L. For each planar graph G with a topological drawing $\mathcal{E}$ and a set S, there is such a drawing if and only if there is an open curve $\mathcal{L}$ that starts and ends in the outer face of $\mathcal{E}$, contains exactly the vertices in S and for each edge e of G, $\mathcal{L}$ either entirely contains e or intersects e at most once [Da +18]. Note that this curve has essentially the same properties as a line L in Γ, except that it is not straight. Thus, we refer to such a curve as a *pseudoline with respect to the embedded graph* G. A surprising recent result is that given a subset S of the vertices and a point set P of size $|S|$, there is a map $\gamma : S \to P$ and a geometric drawing Γ of G that extends γ if there exists a pseudoline $\mathcal{L}$ with respect to G that collects the vertices in S [Duj+19].

In the theoretical part of this thesis, we extend this line of research. For example we prove that given a set of vertices S, it is $\mathcal{NP}$-complete to decide whether there is a pseudoline that collects exactly the vertices in S. On the positive side, we show that under certain conditions the stretchability of an embedded graph and a pseudoline can be generalized to an arrangement of pseudolines.

1.1 Outline and Contribution

This thesis is divided into a practical and a theoretical part. The practical part, Part I, introduces and evaluates algorithms for geometric drawings with a small number of crossings and algorithms that are related to this problem. Part II is concerned with theoretical aspects of planar geometric drawings. In particular, we study whether a topologically embedded planar graph can be stretched to a planar geometric drawing while satisfying specific constraints.

Part I – Crossings in Geometric Drawings

Given a graph $G = (V, E)$ and number $k \in \mathbb{N}$, it is $\mathcal{NP}$-complete to decide whether there is a topological drawing of G that hast at most k crossings [GJ83]. If we require the drawing to be geometric, the problem is $\exists\mathbb{R}$-complete [Bie91]. Thus, in practice it is unlikely that there is an efficient algorithm that computes a geometric drawing with a minimal number of crossings. Chapter 4 and Chapter 5 are concerned with the design and the evaluation of efficient heuristics for this task. In contrast to the geometric setting, there are effective heuristics that minimize the number of crossings in topological drawings [Buc+13]. To profit from these techniques for the geometric setting, we introduce in Chapter 6 techniques to stretch these topological drawings to drawings where the edges are as *straight as possible.* Chapter 7 studies not the number of crossings, but the *crossing angles* in geometric drawings, i.e., the smallest angle incident to a crossing of two edges.

We use descriptive and inferential statistics to evaluate the performance of the implementations. In Chapter 3, we describe the concepts that we use to evaluate the algorithms. As part of this chapter, we introduce the concept of *advantages* which is the base for the inferential statistical test that we use.

Geometric Crossing Minimization

Force-directed algorithms are attributed the property that they tend to produce crossing-free geometric drawings of planar graphs [Kob13]. In Chapter 4, we introduce three heuristics to minimize the number of crossings in geometric drawings. A crucial part is, for a fixed vertex v, to characterize the set P of points p that induces the minimal number of crossings for the edges incident to v when v is moved to p. We show that there is an $O((kn + m)^2 \log(kn + m))$-time algorithm that computes P, where n and m are the number of vertices and edges of G, respectively, and k is the degree of v; see Figure 1.2. In an extensive experimental evaluation we show that for a broad variety of instances the heuristics are able to compute geometric drawings that have about 50% fewer crossings compared to force-directed methods that are implemented in the Open Graph Drawing Framework [Chi+13].

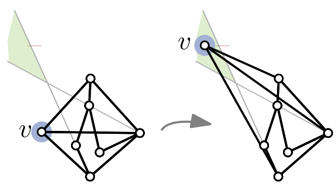


Figure 1.2: Moving the vertex v into the green regions reduces the number of crossings.

A drawback of this approach is that it extensively uses geometric operations that require arbitrarily precise floating point operations. In Chapter 5, we show how a combinatorial tool to compute the dual of a line arrangement allows us to considerably reduce the use of precise floating point operations. On average this yields a speed-up of the computations by a factor of 20. The technique is not restricted to this setting and can be applied to a broad set of geometric operations. Further, we introduce and evaluate a randomized approach to compute a position for a vertex with a small number of crossings. The combination of both techniques allows the computation of geometric drawings with a small number of crossings of graphs with up to 12 000 edges. Note that, in comparison to the evaluated instances in Chapter 4, this increases the number of edges by a factor of 60. The experimental results are complemented by an approximation algorithm with a provable performance guarantee.

Stretching Topological Drawings

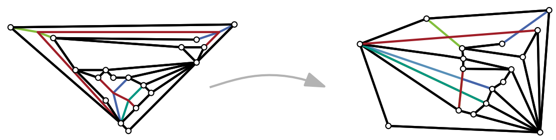


Figure 1.3: The left topological drawing can be stretched to the right geometric drawing.

For topological drawings there are heuristics that successfully minimize the number of crossings in practice [Buc+13]. Unfortunately, it is $\exists\mathbb{R}$-complete to decide whether the drawing can be stretched to a geometric drawing with same number of crossings [BD93]. In Chapter 6, we assume that we are already given a topological drawing with a small number of crossings. The goal is to find a drawing with the same combinatorial properties, i.e., in particular the same number of crossings, and where the edges are as *straight as possible*; see Figure 1.3. We introduce two heuristics for this problem. One heuristic extends a known force-directed method and the other is a geometric approach to this problem. The hypothesis-driven evaluation shows that the geometric approach computes almost-optimal solutions for instances with few crossings per edge. On instances with many crossings per edge the geometric approach computes significantly better results than the force-directed approach.

Crossing-Angle Maximization

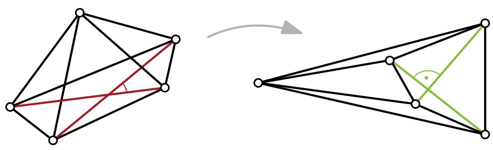


Figure 1.4: The left drawing has a small crossing angle. Ideally, the drawing has a crossing angle of 90°.

In Chapter 7, we study the problem of computing a geometric drawing of a graph that has a large *crossing angle*, i.e., the smallest angle incident to an intersection point of any two crossings edges; compare Figure 1.4. Deciding whether a graph has a geometric drawing with a crossing angle of 90° is an $\mathcal{NP}$-hard problem [ABS12]. In Chapter 7, we introduce a randomized approach that computes geometric drawings with a large crossing-angle. The evaluation shows that the choice of the initial drawing affects the quality of the final drawing. Moreover, we show that the crossing angles of the initial drawings are considerably increased. In particular, this implies that our approach computes drawings with a considerably larger crossing angle than our implementation of known force-directed approaches. The Graph Drawing Contest held during the annual International Graph Drawing and Network Visualization Symposium posed the maximization of the crossing angle as an algorithmic challenge. Our approach is the winning algorithm of the 2017 edition of the graph drawing contest [Dev+18]. The crossing angle in drawings obtained by our approach are larger by a factor of 2 compared to drawings of the competing algorithms.

Part II – Stretching Topological Embeddings with Constraints

In this part, we study geometric drawings of planar graphs. In particular, we investigate whether a topologically embedded planar graph can be stretched to a planar geometric drawing while satisfying given constraints. The problem studied in Chapter 8 is motivated by the crossing-minimization heuristics in Chapter 4. The question is, given a number $K \in \mathbb{N}$, a planar graph G and two vertices s and t, is there a geometric drawing of G such that the edge st can be inserted as straight-line segment with at most K crossings. In Chapter 9 and Chapter 10, we study constraints that restrict the position of the vertices to a set of geometric entities, i.e., a set of disks or lines.

Inserting an Edge into a Geometric Embedding

The problem in Chapter 8 is, given a number $K \in \mathbb{N}$, a planar embedded graph $G = (V, E)$ and a pair $s, t \in V$, is there a planar geometric drawing Γ of G, with the same combinatorial embedding, where the edge st can be inserted as straight-line segment with at most K crossings; see Figure 1.5. If K is the minimum number of crossings on st, where st is an arbitrary curve, then we prove that the problem equivalent to the existence of two specific edge-disjoint paths in a planar graph. This new characterization answers an open question of Eades et al. [Ead+15]. They were only able to prescribe the combinatorial embedding of G but not the choice of the outer face. In contrast to this characterization, our characterization gives conditions for an arbitrary choice of the outer face. Moreover, we show that the problem is fixed-parameter tractable in the number of crossings. In the following, let $\mathcal{E} + st$ be a topological drawing of G such that $\mathcal{E}$ is a planar drawing of G and st has a minimum number of crossings. For a specific graph class, we provide a polynomial-time algorithm

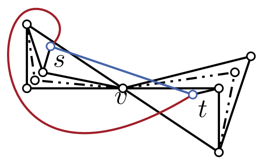


Figure 1.5: The topologically embedded edge st has only two crossings. Every geometric embedding requires at least three crossings.

that decides whether there is a geometric drawing Γ of G with the same outer face and combinatorial embedding as $\mathcal{E}$ such that $\Gamma + st$ has the same number of crossings as $\mathcal{E} + st$. In case that the choice of the outer face is free and the maximum vertex-degree of G is at most 5, we show that there always exists such a drawing $\Gamma + st$. For graphs with a vertex of degree 6, this is not necessarily true. For graphs of maximum degree Δ, we show how to construct a planar geometric drawing Γ of G, such that the number of crossings in $\Gamma + st$ is bounded above by $(\Delta - 2)K$, where K is the number of crossings in $\mathcal{E} + st$.

Planar Graphs on Disks

In Chapter 9, we consider clustered planar graphs, i.e., planar embedded graphs $G = (V, E)$ with a partition $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ of the vertex set. Moreover, we are given a set of disjoint disks $\{D_1, D_2, \ldots, D_k\}$. Let uv be an edge of G such that $u \in V_i$ and $v \in V_j$ with $i \neq j$. We refer to the convex hull of the disks D_i and D_j as a *pipe*; see Figure 1.6. The studied problem asks for a straight-line drawing of G with the same combinatorial embedding and outer face as G, where each vertex in the set V_i lies in the interior of the disk D_i, and each edge lies in the pipe of its corresponding disks and intersects the boundary of each disk at most once. By showing that this problem is $\mathcal{NP}$-hard for unit-sized disks, we answer an open question of Angelini et al. [Ang+14]. In a restricted setting, in which some pipes and disks are not allowed to intersect, we show that each instance admits a geometric drawing where the position of the vertices lie in the interior of the prescribed disks.

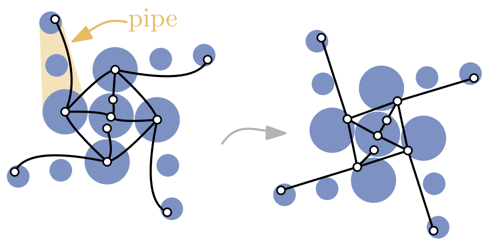


Figure 1.6: A topologically and the corresponding geometrically embedded clustered graphs. The orange region is a pipe.

Aligned Drawings

Let S be a subset of the vertices V of a planar embedded graph G. As stated in the introduction, the graph G has a geometric drawing where the vertices S are positioned on a common line if and only if there is a pseudoline with respect to G that collects all vertices in S [Da +18]; see Figure 1.7. In Chapter 10, we prove that it is $\mathcal{NP}$-complete to decide whether such a pseudoline exists. Fortunately, the problem is fixed-parameter tractable in the number of vertices in S.

In a generalization of the problem we consider, instead of a single pseudoline, a set of pseudolines $\mathcal{A} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k\}$ that is homeomorphic to a line arrangement $A = \{L_1, L_2, \ldots, L_k\}$, where $\mathcal{L}_i$ corresponds to L_i, for $i = 1, 2, \ldots, k$. The problem

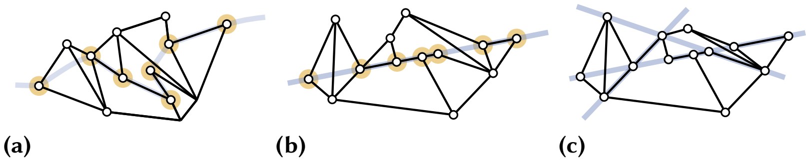


Figure 1.7: (a) A pseudoline that passes through a set S (orange vertices) and (b) a geometric drawing where the vertices lie on a common line. (c) A graph aligned on a line arrangement.

asks for a geometric drawing Γ of G that essentially satisfies that each pseudoline $\mathcal{L}_i$ intersects in the embedding of G the same set of edges as the line L_i in Γ. We show that there are instances that do not have a geometric drawing that satisfies this constraint. Moreover, we prove that if some edge-pseudoline intersection-patterns are forbidden, then every instance has a geometric realization.

2 Terminology

In this section, we introduce terminology and notations that reoccur at several places in this thesis. In particular, we define basic geometric and graph-theoretic concepts, and notions used in the area of graph drawing. We assume familiarity with the computational complexity of decision problems and omit an explanation of the complexity classes $\mathcal{P}$, $\mathcal{NP}$ and related concept as, for example, fixed-parameter tractability. Many geometric problems are only known to be $\mathcal{NP}$-hard, i.e., it is unclear whether the problems are in $\mathcal{NP}$. Indeed, these problems are often $\exists\mathbb{R}$-complete. The *existential theory of the reals ($\exists\mathbb{R}$)* is defined as the set of true sentences of the form $\exists x_1, x_2, \ldots, x_n \in \mathbb{R} f(x_1, x_2, \ldots, x_n)$, where f is a quantifier-free boolean formula over the signature $(0, 1, +, *, <)$ [Sch10]. It is not known whether $\mathcal{NP}$ and $\exists\mathbb{R}$ coincide. For a further overview over the existential theory of the reals we refer to [Sch10].

Graph theoretical notions

An *undirected graph* is a tuple $G = (V, E)$ where $E \subseteq \{\{u, v\} \mid u, v \in V\}$. An element $v \in V$ is a *vertex* and an element $e \in E$ is an *edge*. Formally, an edge is a set $\{u, v\}$ of two vertices u, v. For convenience, we abbreviate $\{u, v\}$ with uv, i.e., $uv = vu$. In a *directed graph* the edge set E is a subset of $V \times V$, i.e., edges are ordered tuples in the form (u, v) with $u, v \in V$. Note that in this case $(u, v) \neq (v, u)$. If the set E is a multiset, then we refer to G as a *multigraph*. An edge vv is a *loop*. A graph without loops is *simple*. If not otherwise stated we assume that a graph is a simple graph.

We denote the number of vertices of G by $n := |V|$ and the number of edges by $m := |E|$. Two vertices u and v are *adjacent* if uv is an edge of G. An edge uv *connects* the two vertices u and v and uv is *incident to u and v*. For a directed graph and a vertex v an edge uv is *incoming for v* and an edge vu is *outgoing for v*. The set of vertices adjacent to a vertex of v is the *(open) neighborhood $N(v)$ of v*, i.e., $N(v) := \{u \mid uv \in E\}$. We denote the set of edges incident to v by $E(v)$. The *degree* of a vertex v is the number of edges incident to v, i.e., $|E(v)|$. For directed graphs, the *in-degree* of v is the number of incoming edges of v and the *out-degree* is the number of outgoing edges of v.

For a (directed) graph $G = (V, E)$, a sequence $p = \langle v_0, v_1, \ldots, v_k \rangle$, for $k \in \mathbb{N}$, of vertices $v_i \in V$ is a *walk*, if $v_{i-1}v_i$ is an edge of G, for $i = 1, \ldots, k$. For a directed graph, p is an *undirected walk*, if $v_{i-1}v_i$ or v_iv_{i-1} is an edge of G. We say that a walk *traverses* an edge $v_{i-1}v_i$, for $i = 1, \ldots, k$, and *contains* the vertex v_j, for $j = 0, 1, \ldots, k$. The vertices v_0 and v_k are the *endpoints of p* and for $k > 1$ and $i = 1, 2, \ldots, k - 1$ the vertex v_i is an *interior vertex of p*.

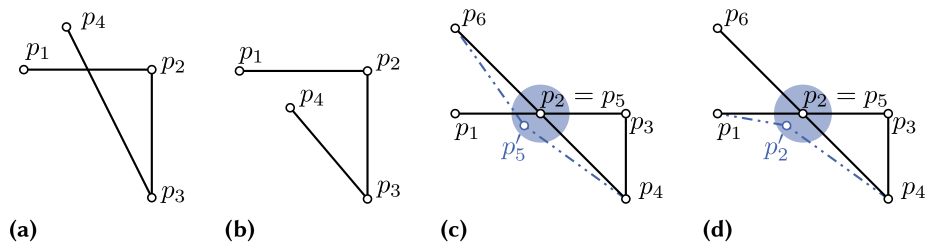


Figure 2.1: The polygonal chain in (a) has an interior intersection and is therefore in contrast to (b) not simple. There is no perturbation of the points of (c) that transforms the polygonal chain to a simple polygonal chain. The polygonal chain in (d) is weakly simple, since there is such a relocation of the points.

A walk that traverses no edge twice is a *path*. A path that contains not vertex twice is a simple path. A walk with $v_0 = v_k$ that does not traverse an edge twice is a *cycle*. A cycle is *simple* if the subpath $\langle v_0, v_1, \dots, v_{k-1} \rangle$ is simple. A simple cycle on three distinct vertices is a *triangle*. A graph is *connected* if for any two vertices u, v there is a path p that has u and v as its endpoints. A graph is *biconnected* if the graphs remains connected after the removal of any vertex.

Geometry

We refer to a tuple $p = (x_p, y_p) \in \mathbb{R}^2$ in the Euclidean plane $\mathbb{R}^2$ as a *point* where x_p and y_p are the *x- and y-coordinates of p*, respectively. We denote the Euclidean distance of two points p and q by $d(p, q)$ or $||p - q||$. The *line* that contains two given points p and q is defined as the set $\{p + r(q - p) \mid r \in \mathbb{R}\}$. The *line segment s from p to q* is the subset of a line that contains all points in between p and q, i.e., $s = \{p + r(q - p) \mid r \in [0, 1]\}$. We will often denote the line segment from p to q as pq. The points p and q are the *endpoints* of the line segment pq. A point u of pq is an *interior point* of pq if it is not an endpoint of pq. The *ray* from p to q is the set $\{p + r(q - p) \mid r \in \mathbb{R}, r \geq 0\}$. The *circle of radius $r \in \mathbb{R}$ with center $c \in \mathbb{R}^2$* is the set $\{p \mid ||c - p|| = r\}$, for $r > 0$. Correspondingly, a *disk of radius r and center c* is the set of points with distance at most r from c, i.e. $\{p \mid ||c - p|| \leq r\}$. A *unit circle (disk)* has radius 1. A line that contains a segment (ray) is the *supporting line of the segment (ray)*. The *cross product of two points p and q* is $p \times q = x_p y_q - y_p x_q$. For a line l directed from p to q, we say a point u *is left of l*, if $(q - p) \times (u - p) < 0$. The point u is *right of l* if $(q - p) \times (u - p) > 0$. Three points p, q and u are *collinear* if there is a line that contains all three points. A (directed) line l divides the plane into two sets, a set H_L of points that are left of l and a set H_R of points that are right of l. We refer to H_L and H_R as the *half-planes of l*. The line l is the *supporting line of H_L and H_R*.

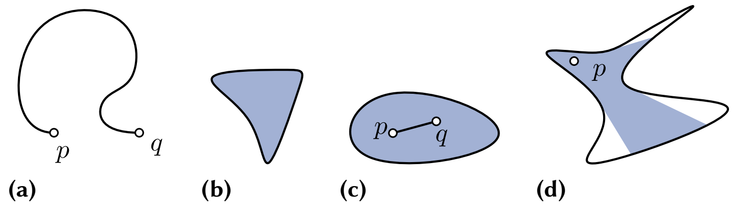


Figure 2.2: Illustration of a (a) Jordan arc, a (b) Jordan curve and its interior (blue), and (c) a convex region. We refer the bounded (blue) region as the interior of the Jordan curve. (c) The blue region is the visibility region of p.

A *polygonal chain* is a tuple of line segments $P = (p_0p_1, p_1p_2, \dots, p_{n-1}p_n)$, for $n \in \mathbb{N}$; see Figure 2.1. A polygonal chain is *simple* if only consecutive segments intersect and then only in their endpoints, where we consider p_0p_1 and $p_{n-1}p_n$ to be consecutive. We call a polygonal chain P *weakly simple* if for any $\epsilon > 0$ there is a relocation (perturbation) of each point p_i of P within a disk of radius ϵ and center p_i such that P becomes simple; compare also [Aki+17]. A polygonal chain is a *polygon* if $p_n = p_0$.

A *Jordan arc* is an injective continuous function $\psi : [0, 1] \to \mathbb{R}^2$. The points $\psi(0)$ and $\psi(1)$ are the *endpoints* of ψ. A set of $M \subset \mathbb{R}^2$ is *path connected*, if for any two points $p \in M$ and $q \in M$, there is a Jordan arc ψ that has p and q as its endpoints and for each $i \in [0, 1]$ the point $\psi(i)$ is in M. Despite the fact that a *connected region* is commonly defined as a region that is not the union of two or more disjoint non-empty open sets, we refer to a path-connected region simply as *connected*. A connected set $M \subset \mathbb{R}^2$ is a *region in* $\mathbb{R}^2$. Let C be a unit circle with center $(0, 0)$. A *Jordan curve* is a injective continuous function $\phi : C \to \mathbb{R}^2$. The famous Jordan curve Theorem states that any Jordan curve ϕ divides the plane into two regions, an *interior* and an *exterior region*. We say the Jordan curve is the *boundary* of these regions. Since a simple polygon is a Jordan curve this applies to simple polygons as well.

Let M be a region. For a given point $p \in M$, a point q is *visible from* p if each point u on the line segment pq is in M. We refer to the set of points that are visible from p as the *visibility region of* p. A region M is *convex* if for any two points $p, q \in M$ each point on the line segment pq is in M.

We refer to a finite set A of lines as a *line arrangement*. A line arrangement divides the plane into a set of regions that we call the *cells of* A; see Figure 2.3. The following definition of pseudoline arrangements is inspired by the definition of the projective plane. Let C be a circle with center $(0, 0)$ and a sufficiently (infinitely) large radius. We call the region bounded by C that does not contain the point $(0, 0)$ *the infinity*. We refer to a Jordan arc that has both its endpoints in the infinity and passes through the disk D_C bounded by C as a *pseudoline*. We refer to a set of pseudolines as an *arrangement of pseudolines* if each pair of pseudolines intersects exactly once and the intersection

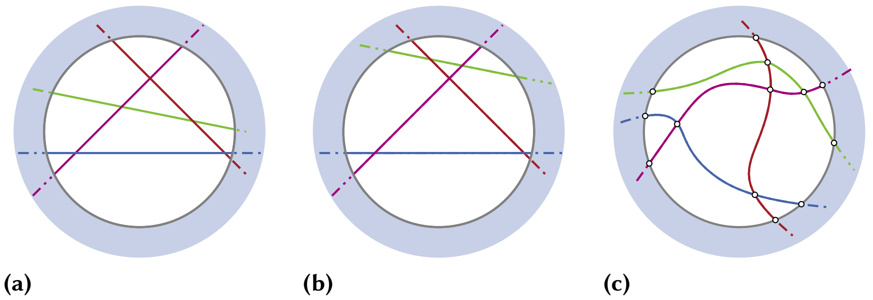


Figure 2.3: (a, b) Two line arrangements that are not homeomorphic. The pseudoline arrangements (b) and (c) are homeomorphic. The points on the intersection in (c) indicate the planar subdivision of the arrangement.

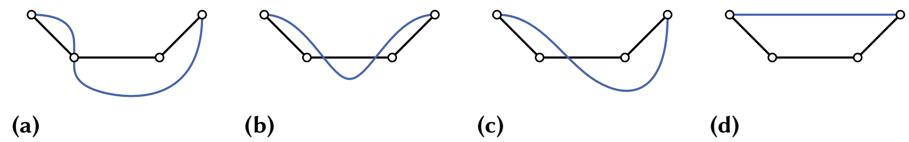


Figure 2.4: (a) A (thick) framework of a 4-cycle but not a drawing. (b) A drawing of graph that is in our notion not a topological drawing. (c) A topological drawing and (d) a geometric drawing.

point is in D_C. Observe that, similar to line arrangements, a pseudolines arrangement divides the disk D_C into cells. A *planar subdivision of a (pseudo-)line arrangement* $\mathcal{A}$ is a planar graph (a formal definition of planar graphs will be given later on) that contains for each intersection of two elements in $\mathcal{A} \cup \{C\}$ a vertex and two vertices are connected by an edge, if their intersections are consecutive on a (pseudo-)line $\mathcal{L}$ of $\mathcal{A}$.

We consider two pseudoline arrangements $\mathcal{A}$ and $\mathcal{A}'$ to be *homeomorphic*, if there is a bijective map $\phi : \mathcal{A} \to \mathcal{A}'$ and each pseudoline in $\mathcal{A}$ and in $\mathcal{A}'$ can be directed such that if a pseudoline $\mathcal{L} \in \mathcal{A}$ intersects pseudolines $\mathcal{L}_1, \mathcal{L}_2, \dots \in \mathcal{A}$ in this order then $\phi(\mathcal{L})$ intersects exactly $\phi(\mathcal{L}_1), \phi(\mathcal{L}_2), \dots \in \mathcal{A}'$ in this order; see Figure 2.3. A pseudoline arrangement $\mathcal{A}$ is *stretchable* if there exists a line arrangement A that is homeomorphic to $\mathcal{A}$.

Drawings of Graphs

A *framework* of a graph $G = (V, E)$ maps each vertex v to a point p_v in the plane and each edge uv to a Jordan arc c_{uv} with p_u and p_v as its endpoints; see Figure 2.4. Observe that in this definition the arc c_{uv} can be a space-filling curve. In order to avoid

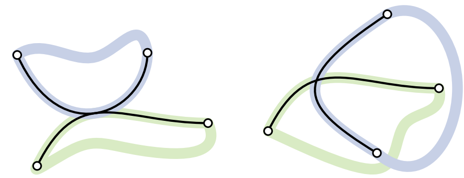


Figure 2.5: The closed curves are a proper extension of the black edges. The edges in (a) are touching edges and the edges in (b) are crossing.

such pathological edge cases, we will only consider *thick frameworks*, i.e., there is an $\epsilon > 0$ such that for each edge uv of G and each point q on c_{uv} and all edges xy of G the circle of radius ϵ and center q has at most two intersection points with c_{xy}, and c_{uv} and c_{xy} have only a finite number of intersection points. A thick framework of G is a *drawing of G* if for any two distinct vertices $u, v \in V$ the points p_u and p_v are distinct and for each edge $uv \in E$ and each vertex $w \in V \setminus \{u, v\}$ the Jordan arc c_{uv} of the edge uv does not contain p_w. For convenience, we do not distinguish between a vertex (edge) and its drawing, i.e., we refer to p_v and c_{uv} simply as v and uv, respectively. Moreover, for the purpose of this thesis, we assume that each pair of distinct edges has at most a single intersection point.

Two edges e_1 and e_2 *intersect* if they have a common intersection point p that is an interior point of e_1 and e_2. Two Jordan curves C_1 and C_2 are a *proper extension of two edges e_1 and e_2* if C_i contains e_i, $C_1 \setminus e_1$ does not intersect e_2, and $C_2 \setminus e_2$ does not intersect e_1; see Figure 2.5. Two intersecting edges e_1 and e_2 *cross* if for every proper extension C_1 and C_2 of e_1 and e_2, each region bounded by C_1 contains an endpoint of e_2 and each region bounded by C_2 contains an endpoint of e_1. Two intersecting edges *touch* if they do not cross. A drawing is a *topological drawing* if each pair of intersecting edges crosses. For convenience, we simply refer to a topological drawing as a drawing. A drawing is a *straight-line drawing* or a *geometric drawing* if each edge is drawn as a straight-line segment.

We denote the number of crossings in a drawing Γ by $\mathrm{cr}(\Gamma)$. The *crossing number* $\mathrm{cr}(G)$ *of a graph G* is the minimum of $\mathrm{cr}(\Gamma)$ over all topological drawings Γ of G. The *rectilinear* or *geometric crossing number* $\overline{\mathrm{cr}}(G)$ *of G* is the minimum of $\mathrm{cr}(\Gamma)$ over all geometric drawings Γ of G. Note that $\mathrm{cr}(G) \leq \overline{\mathrm{cr}}(G)$ for all graphs G.

A drawing is *planar* if no pair of edges crosses. A planar (topological) drawing of G is often called an *embedding* of G. A *combinatorial embedding of G* is a clockwise ordering of the edges around each vertex that corresponds to the clockwise order in a planar drawing of G. A graph G is *planar* if it has a planar drawing, i.e., $\mathrm{cr}(G) = 0$. Note that in this special case, we have by Theorem 1.1 that $\mathrm{cr}(G) = \overline{\mathrm{cr}}(G)$. Let Γ be a planar drawing of G. The drawing partitions the plane into regions, which we call the

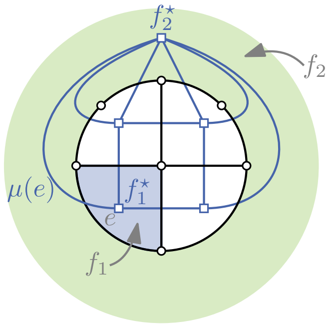


Figure 2.6: A planar drawing $\Gamma^\star$ of a dual graph G of an embedding Γ of G. The green region f_2 indicates the outer face and the blue region f_1 is a inner face.

faces of Γ. There is one unbounded face which we call the *outer face of* G; compare Figure 2.6. A face that is not the outer face is an *interior face*. Each face f is bounded by a set of edges, we refer to these edges as the *boundary of* f. An edge e is incident to a face f if the boundary of f contains e. Two distinct faces f_1 and f_2 are *adjacent* if they are incident to a common edge e. Denote by F_Γ the set of faces of a planar drawing Γ of G. For a planar drawing Γ, a multigraph graph $G^\star = (F_\Gamma, E^\star)$ is a *dual graph* $G^\star$ *of* G if there is a bijective map $\mu : E \longrightarrow E^\star$ with the property that for each $uv \in E$ with $\mu(uv) = f_1 f_2$, f_1 and f_2 are the two faces of G that have uv on its boundary. For a compatible embeddings of G and $G^\star$, we will refer to a vertex of $G^\star$ as a *dual vertex of a face* f *of* G and denote this vertex by $f^\star$. Correspondingly, an edge of $G^\star$ is *dual to an edge of* G. A dual graph naturally comes with a planar drawing $\Gamma^\star$ of $G^\star$, where each vertex $f^\star$ is positioned in the interior of the face f of Γ and for each edge e of G, e and $\mu(e)$ each have exactly one interior intersection point in $\Gamma \cup \Gamma^\star$ and moreover, e and $\mu(e)$ cross in their interior; see Figure 2.6.

A planar drawing is *triangulated* if each face is bounded by a triangle. It is *internally triangulated* if each interior face is bounded by a triangle and the outer face is bounded by a simple polygon. A planar graph is *(internally) triangulated* if it has a (internally) triangulated planar drawing. A triangle in a planar drawing is a *separating triangle* if it is not a face, i.e., each region bounded by the triangle contains a vertex of G in its interior. Let $e = uv$ be an edge that is not incident to a separating triangle. The *contracted graph* G/e is the graph obtained by inserting all edges ux for each neighbor x of v and by removing the vertex v including its incident edges and all multiple edges. Let ψ be a topological drawing of G. We obtain a topological drawing ψ/e of G/e from ψ by routing the new edges ux closely to the drawing of the edges uv and vx. Note that this is always possible, since ψ is a thick framework.

Part I

Crossings in Geometric Drawings

3 Methodology

In this part of this thesis, we consider optimization problems that ask for a drawing $\Gamma^\star$ of a graph G that minimizes (or maximizes) a given function f. The task can be, for example, to compute a drawing with a minimum number of crossings. The problems that we consider in the following sections are $\mathcal{NP}$-hard. For each problem, we develop an algorithm A that computes for a graph G a drawing $A(G)$ with a small (large) value $f(A(G))$ that is not necessarily minimal (maximal). In this setting it is often unclear how to prove a relationship between $f(\Gamma^\star)$ and $f(A(G))$. In order to assess whether the computed values are rather small or large, we take an empirical approach and compare the solutions of different algorithms with each other. Thus, a major contribution of this part are the experimental evaluations of the introduced algorithms. In order to evaluate the quality of the algorithms we use descriptive statistical tools. Moreover, we generalize the well-known *binomial sign test for paired samples* in order to draw statistically significant conclusions. The aim of this section is to familiarize the reader with these concepts. We start in Section 3.1 with the general setting and the statistical tools. We introduce a concept that we refer to as *advantage* of one algorithm over a second. In Section 3.2, we connect this concept to a statistical test. Finally, we describe a framework to formulate a hypothesis that we apply in Chapter 4 and Chapter 6. Without loss of generality, we assume in the following that the optimization problem is a minimization problem, i.e., we ask for a drawing Γ of a graph G that minimizes f.

We used the notion of advantages and the binomial test with advantages in the following publications [BRR17, BRR19, Dem+18, Rad+18, Rad+19, Rad15].

3.1 Descriptive Statistical Tools

In order to evaluate the quality of drawings computed by an algorithm A with respect to an objective function f that maps a drawing to a number in $\mathbb{R}$, we consider a set of graphs $\mathcal{G} := \{G_1, G_2, \dots, G_K\}$, with $K \in \mathbb{N}, K > 0$. For each graph $G_i \in \mathcal{G}$, we apply the algorithm A to G_i and denote the computed drawing by $A(G_i)$. This yields a sequence $f(A(\mathcal{G})) := \langle f(A(G_1)), f(A(G_2)), \dots, f(A(G_K))\rangle$. Without loss of generality, we assume that the values are in non-descending order. The following descriptive statistics of $f(A(\mathcal{G}))$ are examples of functions that can be used to characterize the computed values $f(A(G_i))$.

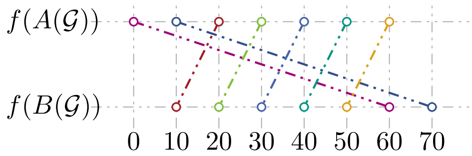


Figure 3.1: Each segment corresponds to a graph in $\mathcal{G}$. The numbers on the x-axis correspond to the values $f(A(G_i))$ and $f(B(G_i))$. Note that the mean of $f(A(\mathcal{G}))$ is smaller than the mean of $f(B(\mathcal{G}))$. But there are only two instances where the values corresponding to A are smaller than the values for B.

The *mean of* $f(A(\mathcal{G}))$ is the value $\sum_{G\in\mathcal{G}} f(A(G))/K$. The *standard deviation of* $f(A(\mathcal{G}))$ is the value $\sqrt{\sum_{G\in\mathcal{G}}\left(f(A(G)) - \overline{f}\right)^2/K}$, where $\overline{f}$ is the mean of $f(A(\mathcal{G}))$. The values $f(A(G_1))$ and $f(A(G_K))$ are the *minimum* and *maximum value of* $f(A(\mathcal{G}))$, respectively. For $q \in (0,1)$, the *(empirical) q-percentile of* $f(A(\mathcal{G}))$ is the value $f(A(G_{\lfloor qK \rfloor +1}))$ if $qK \notin \mathbb{N}$ and otherwise it is $0.5 \cdot (f(A(G_{qK}) + f(A(G_{qK+1}))$. The 0.5-percentile of $f(A(\mathcal{G}))$ is called the *median of* $f(A(\mathcal{G}))$.

These descriptive statistics of $f(A(\mathcal{G}))$ only describe the sequence $f(A(\mathcal{G}))$ but do not necessarily reveal any information about the difference of $f(A(G_i))$ to the optimal value $f(\Gamma^\star)$. Since the problems that we consider in Part I are $\mathcal{NP}$-hard, a practical algorithm O that computes an optimal drawing $\Gamma^\star$ might not be available. As a consequence, we change our perspective and instead of comparing our algorithm A to the algorithm O, we compare A to an established graph drawing algorithm B. Thus, the question becomes whether the value $f(A(G_i))$ is smaller than the value $f(B(G_i))$.

One possibility to approach this question is to compare the descriptive statistics of $f(A(\mathcal{G}))$ to the descriptive statistics of $f(B(\mathcal{G}))$. Note that the conclusion drawn from these descriptive statistics are statements with respect to the aggregated values, e.g., the mean value of $f(A(\mathcal{G}))$ is smaller than the mean value of $f(B(\mathcal{G}))$. As the following example shows, in general, the comparison of the descriptive statistics does not allow a statement about an individual instance G_i.

Consider the sequences $f(A(\mathcal{G}))$ and $f(B(\mathcal{G}))$ given in Figure 3.1. Observe that the mean of $f(A(\mathcal{G}))$ is smaller than the mean of $f(B(\mathcal{G}))$. On the other hand, there are only a few instances G_i on which $f(A(G_i))$ is smaller than the corresponding value $f(B(G_i))$. Thus, depending on whether we compare individual instances to each other or the descriptive statistics, we can draw different conclusions about the performance of A and B.

In the following, we describe one possibility to draw statements about the relationship between $f(A(G_i))$ and $f(B(G_i))$ on the entire set $\mathcal{G}$. If our hypothesis is that A computes drawings of G_i with a smaller value of f than B, for each graph $G_i \in \mathcal{G}$, then we would ideally observe that the inequality $f(A(G_i)) < f(B(G_i))$ is true for all graphs $G_i \in \mathcal{G}$. This notion is very strict in the sense that it requires that the inequality is true *for all* graphs in $\mathcal{G}$. Moreover, even if the inequality is true for all graphs, it does not reveal any information about the difference between $f(A(\mathcal{G}))$ and $f(B(\mathcal{G}))$. We address these two issues with the following model:

> Is there a *large* subset $\mathcal{G}' \subseteq \mathcal{G}$ and a *large* value $\Delta \geq 1$ such that the inequality $f(A(G_i)) \cdot \Delta < f(B(G_i))$ is true for all $G_i \in \mathcal{G}'$?

We introduce the following notion to further formalize this question. Let $\mathcal{G}'$ be a subset of $\mathcal{G}$. We say that A *has an advantage of* $\Delta \geq 1$ *over* B *on* $\mathcal{G}'$ if the inequality $f(A(G_i)) \cdot \Delta < f(B(G_i))$ holds for all $G_i \in \mathcal{G}'$. For a finite set $\mathcal{G}$, we say that a subset $\mathcal{F} \subset \mathcal{G}$ has *relative size at least* $p \in [0, 1]$ if $|\mathcal{F}| \geq p \cdot |\mathcal{G}|$. With this machinery, we can reformulate the previous model as follows:

> For given values $p \in [0, 1]$ and $\Delta \geq 1$, is there a set $\mathcal{G}' \subseteq \mathcal{G}$ of relative size p such that A has an advantage of Δ over B?

Note that the advantage is a relative measure of the distances between the two sets $f(A(\mathcal{G}))$ and $f(B(\mathcal{G}))$. We say that A has an *absolute advantage of* $\Delta \geq 0$ over B on a subset $\mathcal{G}'$ of $\mathcal{G}$ if the inequality $f(A(G_i)) + \Delta < f(B(G_i))$ holds for all graphs $G_i \in \mathcal{G}'$. We will use absolute advantages in case that there are two constants c_1, c_2 such that $f(\Gamma) \in [c_1, c_2]$ for all possible drawings Γ, e.g., if f returns the smallest angle of two crossing edges in Γ.

3.2 Binomial Test with Advantages

The advantage of one algorithm over another describes the relationship between two algorithms with respect to the set $\mathcal{G}$. In case that $\mathcal{G}$ is a subset of a larger family of graphs, for example, planar graphs, it is not necessarily possible to infer properties of the entire set from the observations made on $\mathcal{G}$. In the following, we introduce a statistical test that allows to make statements about the superset of $\mathcal{G}$ in case that $\mathcal{G}$ is a uniform random sample of its superset. We first recite the key ideas behind the binomial test. Further, we connect advantages to the binomial (sign) test. Finally, we propose one possibility to formulate a hypothesis in our setting.

Binomial Test

In this section, we give a short introduction to the binomial test. The following description is based on Sheskin [She03]. The binomial test assesses the likelihood of whether a binary sequence $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$, with $a_i \in \{0, 1\}$, for $n \in \mathbb{N}$, is the result of an experiment where the outcome 1 has probability at least $\pi \in [0, 1]$ and the outcome 0 has probability at most $1 - \pi$.

By $|1|_\sigma$ we denote the number of occurrences of 1 in σ, i.e., $|1|_\sigma = \sum_{i=1}^{n} a_i$.

Let $\pi \in [0,1]$ be a fixed value. Consider a sequence $\sigma_\pi = \langle b_1, b_2, \ldots, b_n \rangle$ that is the result of a binomial trial, for $n \in \mathbb{N}$, i.e., for each $i = 1, 2, \ldots, n$, $b_i = 1$ with probability π and $b_i = 0$ with probability $1 - \pi$. We say that π is the *probability of* σ_π. Let $x \in \mathbb{N}$ with $x \leq n$. The probability that a sequence σ_π has the property that $|1|_{\sigma_\pi} = x$ is $P_\pi^=(x) := \binom{n}{x}\pi^x(1-\pi)^{n-x}$. The probability that $|1|_{\sigma_\pi} \geq x$ is $P_\pi^\geq(x) = \sum_{i=x}^{n} P_\pi^=(i)$.

We now turn back to the initial question whether a binary sequence σ is the result of an experiment where the outcome of 1 and 0 has probability at least p and at most $1-p$, respectively. In this setting the true probability p of σ is unknown. For this purpose, we formulate a *null hypothesis* and an *alternative hypothesis*. The null hypothesis is that the probability π of a given sequence σ is at most a value $p \in [0,1]$. Conversely, the alternative hypothesis is that the probability π of σ is larger than p, i.e., σ is indeed the result of the assumed experiment. The binomial test quantifies the probability that the null hypothesis is true even though we decided to *accept* the alternative hypothesis as the truth. This wrong decision is often called a *type 1 error*. Note that if the probability $P_p^\geq(|1|_\sigma) \leq \alpha$ for a value $\alpha \in (0,1)$, then the probability that the null hypothesis is true, is at most α.

We use the following terminology. For a fixed *significance level* $\alpha \in (0,1)$, we *reject* the null hypothesis, if $P_p^\geq(|1|_\sigma) \leq \alpha$. Thus, if we reject the null hypothesis there is only a small chance that we made a type 1 error, i.e., the null hypothesis actually is true. In case that we reject the null hypothesis, we say that we *accept the alternative hypothesis at significance level of* α.

Binomial Test with Advantages

In this section we connect the concept of advantages to the binomial test. Note that the *binomial sign test with advantages* is a generalization of the binomial sign test for paired samples; compare Sheskin [She03].

A set of graphs $\mathcal{G}$ might be too large to apply the algorithms A and B to each graph in $\mathcal{G}$, e.g., it can be computationally too expensive or the size of set $\mathcal{G}$ is simply not finite. Nevertheless, it is desirable to draw reliable statements about the relationship of A and B on the set $\mathcal{G}$. In particular, we study the following alternative hypothesis: For fixed values $p \in [0,1]$, $\Delta \geq 1$ and a graph G drawn uniformly at random from $\mathcal{G}$, the probability π that the inequality $f(A(G)) \cdot \Delta < f(B(G))$ is true is at least p. The respective null hypothesis is that the inequality is true with probability at most p. Observe that this corresponds to an experiment with exactly two outcomes, i.e., the inequality $f(A(G)) \cdot \Delta < f(B(G))$ is either true or false. Thus, we can apply the binomial test as follows.

For $k \in \mathbb{N}$, let $\mathcal{G}' = \{G_1, G_2, \ldots, G_k\}$ be a finite set of graphs drawn uniformly at random from $\mathcal{G}$. Let $\sigma = \langle a_1, a_2, \ldots, a_k \rangle$ be a binary sequence, where $a_i = 1$ if the inequality $f(A(G_i)) \cdot \Delta < f(B(G_i))$ is true and $a_i = 0$, otherwise. The previous alternative hypothesis can be reformulated as that the probability of σ is at least p. Hence, we can use the binomial test to test our hypothesis. In particular, we reject the null hypothesis phrased in the previous paragraph if we reject the hypothesis that the probability of σ is at most p. As before, we *accept the alternative hypothesis at a significance level* $\alpha \in (0, 1)$, if we do not reject the null hypothesis. If we *accept* the alternative hypothesis that A has an advantage of Δ over B with probability p, then it is unlikely that the probability that A has an advantage of Δ over B is at most p. Note the subtle difference to the case that the alternative hypothesis is indeed true. Then we expect that for a finite subset $\mathcal{G}' \subset \mathcal{G}$ drawn uniformly at random that there is a subset $\mathcal{G}'' \subset \mathcal{G}'$ of relative size at least p such that A has an advantage of Δ over B on $\mathcal{G}''$.

Formulating and Testing Hypotheses

In contrast to the hypothesis of the binomial test the hypothesis of the binomial test with advantages bases on two values, i.e., the value π and the advantage Δ, which depend on each other. We introduce one possibility to formulate a hypothesis in this setting.

Let $\mathcal{G}' \subseteq \mathcal{G}$ be a finite random sample and let $q \in [0, 1]$. We partition $\mathcal{G}'$ into two sets $\mathcal{G}_{\text{test}}$ and $\mathcal{G}_{\text{verify}}$. Where $\mathcal{G}_{\text{test}}$ has relative size q with respect to $\mathcal{G}'$ and $\mathcal{G}_{\text{verify}}$ has relative size $1 - q$. For a given value $p \in [0, 1]$, if A has an advantage over B on a subset of relative size p of $\mathcal{G}_{\text{test}}$, we compute the maximum value Δ such that A has an advantage of Δ over B on a subset of $\mathcal{G}_{\text{test}}$ of relative size p. Note that, if we choose a value Δ' that is smaller than Δ, we increase the chance that A has an advantage of Δ' over B on a subset of relative size p of $\mathcal{G}_{\text{verify}}$. Thus, in order to increase the likelihood that we can reject the null hypothesis, we ask whether A has an advantage of $\min(1, c \cdot \Delta)$ over B, for $c \in (0, 1)$. In this thesis we use $c = 0.75$. In case that we do not reject the null hypothesis, we say that the advantage is *significant*.

4 Geometric Crossing Minimization

In this chapter we consider the *geometric crossing minimization problem*, i.e., we seek a straight-line drawing Γ of a graph $G = (V, E)$ with a small number of edge crossings. Crossing minimization is an active field of research [ÁFS13, Buc+13]. While there is a lot of work on heuristics for topological drawings, these techniques are typically not transferable to the geometric setting. We introduce and evaluate three heuristics for geometric crossing minimization. The approaches are based on the primitive operation of moving a single vertex to its crossing-minimal position in the current drawing Γ, for which we give an $O((kn+m)^2 \log(kn+m))$-time algorithm, where k is the degree of the vertex and n and m are the number of vertices and edges of the graph, respectively. In an experimental evaluation, we demonstrate that our algorithms compute straight-line drawings with fewer crossings than energy-based algorithms implemented in the Open Graph Drawing Framework [Chi+13] on a varied set of benchmark instances. Additionally, we show that the difference of the number of crossings of topological drawings computed with the edge insertion approach [Buc+13, CH16] and the number of crossings in straight-line drawings computed by our heuristic is relatively small. The final experiments are evaluated with a statistical significance level of $\alpha = 0.05$.

The research of this chapter was initiated in the Master thesis of Klara Reichard [Rei16]. This chapter is based on joint work with Klara Reichard, Ignaz Rutter and Dorothea Wagner [Rad+18, Rad+19].

4.1 Introduction

The empirical study of Purchase et al. [PCJ96] indicates that a drawing of a graph with a small number of crossings is easier to comprehend than a drawing of the same graph with a large number of crossings. Consequently, the minimization of crossings has received considerable attention in theory and in practice; the bibliography of Vrt'o is an impressive list of over 700 references [Vrt14]. A *topological drawing* of a graph is a drawing where each edge is a Jordan arc and in a *straight-line drawing* each edge is restricted to be a straight-line segment. The *crossing number* $\mathrm{cr}(G)$ of a graph G is the minimum crossing number of all possible topological drawings of G. The *rectilinear crossing number* $\overline{\mathrm{cr}}(G)$ of G is the minimum number of crossings over all possible straight-line drawings of G. Indeed, there is a family of graphs with a constant crossing number but an unbounded rectilinear crossing number [BD93]. Moreover, there is a difference in the algorithmic complexity of the respective minimization problem. The minimization of the crossing number is $\mathcal{NP}$-complete [GJ83]. For the minimization of the rectilinear crossing number only $\mathcal{NP}$-hardness is known, more precisely the problem is $\exists\mathbb{R}$-complete [Bie91]. Due to these gaps, we can either insist on a small number of crossings or on straight-line edges. In case of topological drawings iteratively inserting edges into a (planar) graph with a small number of crossings proved to be effective in practice [Buc+13, CH16]. Unfortunately, deciding whether there is a straight-line drawing homeomorphic to a given drawing is $\exists\mathbb{R}$-complete [Sch10, Sho91]. Based on the topological drawings with a small number of crossings, in Chapter 6, we heuristically straighten the edges. In general it is not possible to transfer the results on topological drawings to the geometric setting. Thus, if we insist on straight-line drawings, there is need for a geometric approach.

Several surveys [ÁFS13, Buc+13] show that the estimation of the (rectilinear) crossing number of complete graphs has received considerable attention. Most recently Fox et al. [FPS16] introduced an $n^{2+o(1)}$-time algorithm that computes a straight-line drawing of a graph G with at most $\overline{\text{cr}}(G) + o(n^4)$ pairs of crossing edges. This is a $1 + O(1)$ approximation for dense graphs but rather of theoretical interest for sparse graphs. A considerable number of known upper bounds for the rectilinear crossing number of the complete graphs K_n for $n \leq 100$ [Aic19] is due to Fabila-Monroy and López [FL14].

Energy-based algorithms are a common way to compute straight-line drawings of arbitrary graphs. For a detailed description we refer to the survey of Kobourov [Kob13]. Energy-based algorithms are often designed to compute drawings with e.g. uniform edge length or small stress. Kobourov claims that these algorithms tend to produce crossing-free drawings for planar graphs. The force-directed approach by Davidson and Harel [DH96] actively reduces the number of crossings among other optimization criteria. Apart from that we are not aware of any algorithms that compute straight-line drawings with a small number of crossings.

Contribution and Outline. Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E and let Γ be a straight-line drawing of G. For a vertex $v \in V$ and a point $p \in \mathbb{R}^2$ we denote by $\Gamma[v \mapsto p]$ the straight-line drawing obtained from Γ by moving v to the point p. Based on the assumption that we are able to compute a drawing $\Gamma[v \mapsto p^\star]$ with a small number of crossings, we introduce in Section 4.2 three heuristics in order to compute drawings with few crossings. In Section 4.3 we show that a drawing $\Gamma[v \mapsto p^\star]$ with a minimum number of crossings can be computed in $O((kn + m)^2 \log(kn + m))$ time for a graph with n vertices, m edges, and a vertex v of degree k. In Section 4.4 we experimentally evaluate our algorithms and show that we achieve fewer crossings than energy-based algorithms implemented in the *Open Graph Drawing Framework* [Chi+13] with a statistical significance of $\alpha = 0.05$. Additionally, we compare our algorithm to topological drawings with a small number

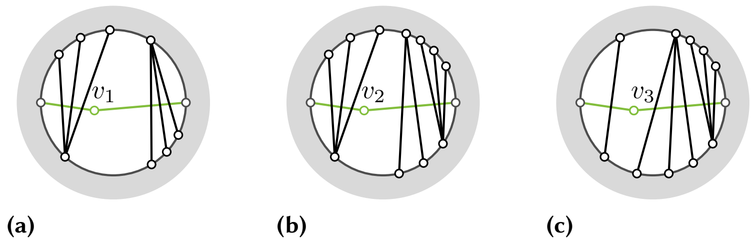


Figure 4.1: Assume that structures in (a)-(c) are substructures of a common drawing Γ. Depending on the function $\mathrm{cr}_x(\Gamma, \cdot)$ the vertices are moved in a different ascending order. For $x = \textsc{Log}$ we have that $\mathrm{cr}_{\textsc{Log}}(\Gamma, v_3) < 3.81 \leq \mathrm{cr}_{\textsc{Log}}(\Gamma, v_1) = 4 < 4.5 < \mathrm{cr}_{\textsc{Log}}(\Gamma, v_2)$. For $x = \textsc{Sum}$ we have that $\mathrm{cr}_{\textsc{Sum}}(\Gamma, v_1) = 6 < \mathrm{cr}_{\textsc{Sum}}(\Gamma, v_3) = 7 < \mathrm{cr}_{\textsc{Sum}}(\Gamma, v_2) = 8$. For $x = \textsc{Sq}$, we have that $\mathrm{cr}_{\textsc{Sq}}(\Gamma, v_1) = 18 < \mathrm{cr}_{\textsc{Sq}}(\Gamma, v_2) = 34 < \mathrm{cr}_{\textsc{Sq}}(\Gamma, v_3) = 37$.

of crossings. We show that there is only a small gap between the number of crossings in topological and straight-line drawings of our benchmark instances. Throughout the remainder of this chapter, a *drawing* of a graph is a straight-line drawing.

4.2 A Framework for Rectilinear Crossing Minimization

Let v be a vertex of the graph $G = (V, E)$ and let Γ be a drawing of G. Recall that the drawing $\Gamma[v \mapsto p]$ is obtained from Γ by moving v to p. Assume that we are able to efficiently compute a position $p^\star$ so that the number of crossings is minimized over all drawings $\Gamma[v \mapsto p], p \in \mathbb{R}^2$. With this operation at hand, several possibilities arise to compute a drawing of G with a small rectilinear crossing number. We introduce three approaches. The *vertex movement approach* iteratively moves the vertices in some order to their locally optimal position. The *vertex insertion approach* starts from a large induced planar subgraph and inserts vertices at their locally optimal position. The *edge insertion approach* starts with a maximal planar subgraph and iteratively inserts edges into the drawing and locally modifies the drawing to reduce the number of crossings.

4.2.1 Vertex Movement Approach

Let $S = \langle v_1, v_2, \ldots, v_k \rangle$, $k \in \mathbb{N}$, be a sequence of vertices of G and let Γ_0 be an arbitrary straight-line drawing of G. The drawing Γ_i is obtained from Γ_{i-1} by moving vertex v_i to its locally optimal position.

The number of crossings in Γ_n may depend on the order S. Hence, we introduce the following possibilities to choose S. As a baseline we use a random permutation of V for S. We refer to this sequence as Random. To obtain other sequences S, we order the vertices V in descending or ascending order with respect to the number of crossings of v in the initial drawing Γ_0 of G. Denote by $E(v)$ the set of edges incident to v, and by $\mathrm{cr}(\Gamma, e)$ the number of crossings of an edge e in the drawing Γ. We propose the following ways to count the number of crossings incident to a vertex v. Figure 4.1 illustrates that these can yield different orders of the same vertex set.

$$\mathrm{cr}_{\textsc{Log}}(\Gamma, v) = \sum_{e \in E(v)} \log(\mathrm{cr}(\Gamma, e) + 1) \tag{4.1}$$

$$\mathrm{cr}_{\textsc{Sum}}(\Gamma, v) = \sum_{e \in E(v)} \mathrm{cr}(\Gamma, e) \tag{4.2}$$

$$\mathrm{cr}_{\textsc{Sq}}(\Gamma, v) = \sum_{e \in E(v)} \mathrm{cr}(\Gamma, e)^2 \tag{4.3}$$

4.2.2 Vertex Insertion Approach

In the vertex insertion approach we identify a subset $V' \subset V$ so that the induced subgraph G_P of $V \setminus V'$ is a planar subgraph of G. Starting from a planar drawing Γ_p of G_p we iteratively insert the vertices in V' at their locally optimal position into Γ_p. Since the respective decision problem of deciding whether there is set V' of at most k vertices is known to be $\mathcal{NP}$-complete [KD79, LY80], we take the following greedy approach.

Let Γ be a non-planar straight-line drawing of G. Let $T' = \langle v_1, v_2, \dots v_n \rangle$ be an ascending (or descending) order of the vertices of G with respect to their number of crossings cr_x in Γ with $x = \textsc{Log}, \textsc{Sum}, \textsc{Sq}$. Let i be the smallest index such that the sub-drawing Γ_i of Γ induced by the vertices $v_i, \dots, v_n$ is planar, i.e., the vertices $V' = \{v_j \mid j = 1, 2, \dots, i-1\}$ are removed from Γ. We obtain a drawing Γ_j from Γ_{j+1} by inserting v_j at its locally optimal position in Γ_{j+1} for $j = 1, \dots i-1$.

4.2.3 Edge Insertion Approach

The following heuristic is inspired by the topological edge-insertion algorithm introduced by Gutwenger et al. [GMW05]. We start with a maximal planar subgraph of G and iteratively reinsert edges e into the previous drawing. We modify each drawing so that we can add the edge e with a small number of crossings. It is $\mathcal{NP}$-complete to decide whether there is a set E' of k edges such that the graph $G' = (V, E \setminus E')$ is planar [GJ79]. Fortunately, there are exact and heuristic approaches known [CHW18, JLM98]. For further details we refer to Section 4.4.6.

Note that we assume all vertices to be in general position. More formally, let $e = uv$ be an edge of a graph G and Γ_{-e} be a straight-line drawing of $G - e$. We obtain a drawing Γ_{+e} of G by inserting e into Γ_{-e} as a straight-line segment. In the following we discuss strategies to locally modify the drawing Γ_{+e} to obtain a drawing Γ with a small

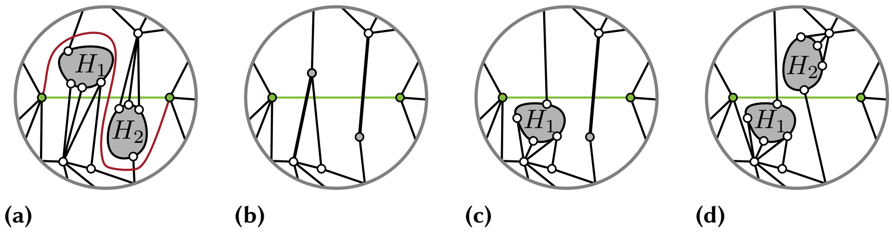


Figure 4.2: (a) A crossing minimal curve C_{uv} with dense Subgraphs H_1 and H_2. (b) Graph with the contracted Subgraphs H_1 and H_2. (c) H_1 unpacked. (d) H_1 and H_2 unpacked.

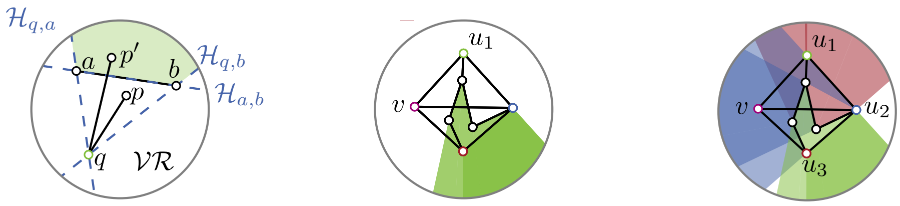
number of crossings. Let C_{uv} be a *crossing minimal curve* from u to v, i.e., a Jordan arc in Γ_{-e} with u and v as its endpoints, only intersecting edges in its interior and with a minimal number of edge crossings; see Figure 4.2a. Ideally, we can rearrange Γ_{-e} such that the edges crossed by e in Γ_{+e} are the same as the edges crossed by C_{uv}. Note, that this problem is closely related to the stretchability of pseudolines which is known to be $\exists\mathbb{R}$-complete [Sho91].

Endpoint. The Endpoint strategy solely moves the endpoints u and v of the inserted edge e in an arbitrary order to their locally optimal position.

Crossed Neighborhood. For a vertex x and an edge e, denote the number of edges xy that cross e in Γ_{+e} by $\mathrm{cr}(\Gamma_{+e}, e, x)$. Let C_e be the set of vertices with $\mathrm{cr}(\Gamma_{+e}, e, x) > 0$. In addition to the endpoints of e, the Crossed Neighborhood strategy moves the vertices in C_e in an order depending on the crossing number cr_a, a = Log, Sum, Sq to their locally optimal position.

Subgraph. Let C_{uv} be a *crossing-minimal curve* from u to v in Γ_{-e} and let E' be the edges crossing C_{uv}. Let R be the (not necessarily simple) region enclosed by e and C_{uv}; see Figure 4.2. The region R partitions G into a set of subgraphs $H_1, H_2, \ldots, H_k$ of G with drawings $\Gamma_{H_1}, \Gamma_{H_2}, \ldots, \Gamma_{H_k}$ in the interior of R. Let E_j be the set of edges uv with $u \in V \setminus V(H_j)$ and $v \in V(H_j)$.

Let Γ_0 be the drawing obtained from Γ_{+e} by contracting every subgraph H_j to a vertex c_j and placing the vertex in the barycenter of the vertices of H_j. In order to obtain a drawing Γ_j from Γ_{j-1}, consider a connected region f_j such that moving the vertex c_j within f_j in Γ_{j-1} yields the same number of crossings, i.e., $\mathrm{cr}(\Gamma_{j-1}[c_j \mapsto p]) = \mathrm{cr}(\Gamma_{j-1}[c_j \mapsto p'])$ for every pair of points $p, p' \in f_j$. Let $f_j^\star$ be the region containing the crossing minimal position $p_j^\star$ of the vertex c_j in the drawing Γ_{j-1} (we prove the existence of such a region in Section 4.3). We obtain a drawing Γ_j by placing a scaled drawing Γ_{H_j} in the interior of $f_j^\star$ and reinserting the edges E_j and deleting c_j and its edges. This operation can introduce new crossings of the edges E_j with Γ_{H_j}. We resolve these crossings by repositioning every vertex $w \in V(H_j)$ to its locally optimal position with respect to the drawing Γ_j.



(a) **(b)** **(c)**

Figure 4.3: The figures highlight the complements of the visibility regions. (a) The visibility region $\mathcal{VR}(q, s)$ of a point q and a segment $s = \mathcal{S}[a, b]$. (b) All regions $\mathcal{VR}(u_1, e)$ for a neighbor u_1 of v. (c) All regions $\mathcal{VR}(u_j, e)$ for all neighbors u_j of a vertex v.

4.3 Locally Optimal Vertex Movement

Let Γ be a drawing of a graph G and v be a vertex of G. The algorithms introduced in Section 4.2 are based on the assumption that we can efficiently compute a position $p^\star$ so that the number of crossings in the drawing $\Gamma[v \mapsto p^\star]$ is minimized. In this section we show that this is possible in $O((kn + m)^2 \log(kn + m))$ time for a degree-k vertex.

In the following we refer to the edges incident to the vertex v as *active*. The remaining edges are called *inactive*. Let uv be an active edge and let e be an inactive edge. We characterize the set of points p such that moving v to p introduces a crossing between uv and e. Based on the resulting region, we define an arrangement $A(\Gamma, v)$. Moving the vertex v within a face of this arrangement does not change the number of crossings. Thus computing an optimal position $p^\star$ reduces to finding a particular face in $A(\Gamma, v)$.

The mentioned characterization is based on the notion of *visibility*. Let $q \in \mathbb{R}^2$ be the position of u and let $s = \mathcal{S}[a, b] \subset \mathbb{R}^2$ be a closed segment between two points a and b. Let $\mathcal{VR}(q, s) \subset \mathbb{R}^2$ be the *visibility region* of q with respect to s, i.e., the set of points $p \in \mathcal{VR}(q, s)$ so that the segments s and $\mathcal{S}[q, p]$ do not intersect. Clearly, $\mathcal{VR}(q, s)$ is the union of three half-planes $\mathcal{H}_{q,a}, \mathcal{H}_{q,b}$ and $\mathcal{H}_{a,b}$ as depicted in Figure 4.3a. We denote the boundary of $\mathcal{VR}(q, s)$ by $\mathcal{BD}(q, s)$. Let $A(\Gamma, v)$ be the arrangement obtained from intersecting the boundaries $\mathcal{BD}(u, e)$ for all pairs of active edges $uv \in E$ and inactive edges e; see Figure 4.3b and Figure 4.3c. We show that moving the vertex v within a face of this arrangement does not change the number of crossings in the drawing $\Gamma[v \mapsto p]$. Thus it is sufficient to compute this arrangement and determine the face $f^\star$ inducing the smallest number of crossings. To avoid special cases, we assume that all vertices are in general position.

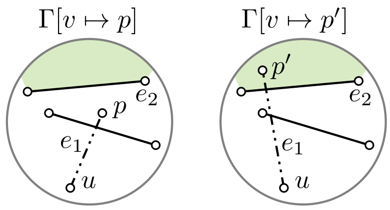


Figure 4.4: Moving the vertex v within a face of $A(\Gamma, v)$ does not change the number of crossings. Illustration for the contradiction.

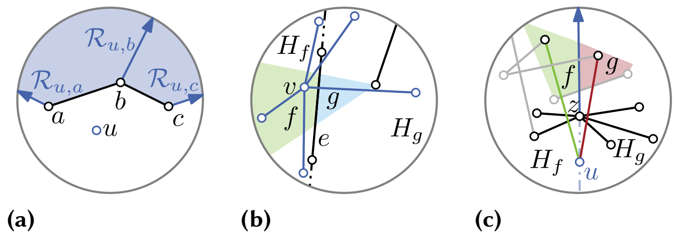


Figure 4.5: (a) The boundary $\mathcal{BD}(q, ab)$ is bounded by the two rays $\mathcal{R}_{q,a}, \mathcal{R}_{a,b}$ and the edge ab. The ray $\mathcal{R}_{q,b}$ lies on the boundary of $\mathcal{BD}(q, ab)$ and $\mathcal{BD}(a, bc)$. (b) The faces f and g share a segment incident to an edge e. (c) The faces f and g share a segment on $R_{q,u}$.

Lemma 4.1. *Let $G = (V, E)$ be a graph with a vertex $v \in V$ and let Γ be a straight-line drawing of G. Let f be a face of $A(\Gamma, v)$, and let p and p' be two points in the interior of f. Then p and p' have the same crossing number, i.e.,* $\operatorname{cr}(\Gamma[v \mapsto p]) = \operatorname{cr}(\Gamma[v \mapsto p'])$.

Proof. For the sake of a contradiction, assume that there are two distinct points p and p' in the interior of f, so that $\operatorname{cr}(\Gamma[v \mapsto p]) < \operatorname{cr}(\Gamma[v \mapsto p'])$. This implies that there is a pair of an active edge e_1 and an inactive edge e_2 that cross in $\Gamma[v \mapsto p']$ but not in $\Gamma[v \mapsto p]$; see Figure 4.4. Thus p' is not contained in $\mathcal{VR}(v, e_2)$ but p is. This contradicts the assumption that both p and p' lie in the interior of the same face of $A(\Gamma, v)$. □

Due to Lemma 4.1 it is sufficient to consider only one point p in the interior of a face f in order to evaluate the crossing number $\operatorname{cr}(\Gamma[v \mapsto q])$ for an arbitrary point q in f. Thus, in the following we denote with $\Gamma[v \mapsto f]$ a drawing, where v is moved to an arbitrary point in f.

Theorem 4.2. *Let Γ be a straight-line drawing of a graph $G = (V, E)$ and let v be a degree-k vertex of G. A point $p^\star \in \mathbb{R}^2$ with the property that $\operatorname{cr}(\Gamma[v \mapsto p^\star]) = \min_{q \in \mathbb{R}^2} \operatorname{cr}(\Gamma[v \mapsto q])$ can be computed in $O((kn + m)^2 \log(kn + m))$ time.*

Proof. The proof relies on the following claims.

Claim 1. *The arrangement $A(\Gamma, v)$ has $O((kn+m)^2)$ vertices. Moreover, it can be computed in $O((kn+m)^2 \log(kn+m))$ time.*

For each active edge uv we obtain $O(m)$ visibility regions. The boundary $\mathcal{BD}(q, ab)$ with respect to an edge ab can be represented by two rays $\mathcal{R}_{q,a}$, $\mathcal{R}_{q,b}$ and the edge ab, see Figure 4.5a. Observe that the two edges ab, bc share a common ray $\mathcal{R}_{q,b}$. Thus, there are in total $O(kn+m)$ geometric entities ($O(kn)$ rays and $O(m)$ edges) with at most $O((kn+m)^2)$ intersections. Thus, we can compute $A(\Gamma, v)$ with a sweep-line algorithm [BO79] in $O((kn+m)^2 \log(kn+m))$ time.

Claim 2. *For all faces f and g of $A(\Gamma, v)$ that share a segment s the values $\Delta_{f,g}$ such that $\mathrm{cr}(\Gamma[v \mapsto g]) = \mathrm{cr}(\Gamma[v \mapsto f]) + \Delta_{f,g}$ can be computed in $O((kn+m)^2)$ time.*

We distinguish whether the segment s lies on an edge e or on a ray $\mathcal{R}_{u,z}$ for a neighbor u of v and $z \in V$. In both cases we show that the value $\Delta_{f,g}$ is equal for all pairs of faces f, g that share a segment on e or $\mathcal{R}_{u,z}$, respectively.

First, consider the case that s lies on an edge $e = xy$ in Γ; see Figure 4.5b. Denote by H_f and H_g the half-planes of the line that contains s such that H_f contains f and H_g contains g. Let p_f and p_g be points in f and g, respectively, that are sufficiently close to s. Note that, since we assume the vertices to be in general position, there is no vertex $z \neq x, y$ that lies on the line that contains s, Thus, an edge uv and s cross in $\Gamma[v \mapsto p_f]$ if and only if $u \in H_g$. Correspondingly, uv and s cross in $\Gamma[v \mapsto p_g]$ if and only if $u \in H_f$. Let n_f and n_g be the number of vertices incident to v contained in H_f and H_g, respectively. Hence, we have that $\Gamma[v \mapsto p_g] = \Gamma[v \mapsto p_f] + n_f - n_g$. Due to Lemma 4.1 it follows that $\Delta_{f,g} = n_f - n_g$. Moreover, the number n_g and n_f are equal for all segments on e, i.e., it is sufficient to compute n_g and n_f with respect to Γ and not for each segment s in $A(\Gamma, v)$. Overall the counting requires $O(km)$ time and mapping these values to differences $\Delta_{f,g}$ requires additional time linear in the size of the arrangement, i.e., $O((kn+m)^2)$ time.

Second, consider the case that s lies on a ray $\mathcal{R}_{u,z}$, i.e., the ray originates in a vertex z and the direction is determined by a neighbor u of v; see Figure 4.5c. As before, let H_f and H_g be the half-planes that contain f and g, respectively. Since all vertices lie in general position, we have the following. Each edge wv with $w \neq u$ crosses the same edges in $\Gamma[v \mapsto f]$ as in $\Gamma[v \mapsto g]$. The edges uv and xy cross in $\Gamma[v \mapsto f]$, with $z \neq x, y$, if and only if uv and xy cross in $\Gamma[v \mapsto g]$. Moreover, the edges uv and xz cross in $\Gamma[v \mapsto f]$ if and only if x lies in H_f. Correspondingly, uv and xz cross in $\Gamma[v \mapsto g]$ if and only if x lies in H_g. Let n'_f and n'_f the number of neighbors of z that lie in H_f and H_g, respectively. Thus, $\Delta_{f,g} = n'_g - n'_f$.

The values n'_f and n'_g can be computed in $O(d_u)$ time, where d_u is the degree of u. Since all differences $\Delta_{f,g}$ are equal for all pair of faces f, g that have a common segment that lies on $\mathcal{R}_{u,z}$, all differences can be computed in $O(\sum_{q \in N_v} \sum_{u \in V} d_u) = O(km)$ time.

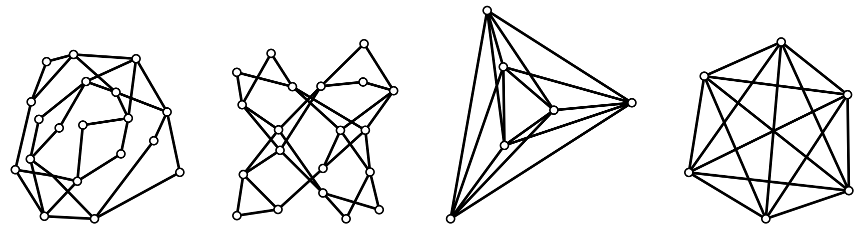


Figure 4.6: (a, c) Example drawings computed by our Edge Insertion heuristic with a repositioning with PrEd. (b, d) Drawings computed with Stress. (a, b) North graph 20.47. (c, d) K_6. Number of crossings: (a) 5, (b) 11, (c) 3, (d) 15.

In time linear in the size of $A(\Gamma, v)$ these values can be mapped to segments in $A(\Gamma, v)$. This finishes the proof of the second claim.

In the following v_f denotes the dual vertex of a face f of $A(\Gamma, v)$.

Claim 3. *Let s and t be two faces of $A(\Gamma, v)$ and let s contain v in its interior. Let Π be a simple path from v_s to v_t in the dual graph of $A(\Gamma, v)$. Then* $\mathrm{cr}(\Gamma[v \mapsto t]) = \mathrm{cr}(\Gamma) + \sum_{(v_f, v_g) \in \Pi} \Delta_{f,g}$.

Since s contains v in its interior, the number of crossings in Γ and $\Gamma[v \mapsto s]$ coincide, i.e., $\mathrm{cr}(\Gamma[v \mapsto s]) = \mathrm{cr}(\Gamma)$. Secondly, for two adjacent faces f and g, we can express the number of crossings in $\Gamma[v \mapsto g]$ depending on the number of crossings in $\Gamma[v \mapsto f]$ and $\Delta_{f,g}$, i.e., $\mathrm{cr}(\Gamma[v \mapsto g]) = \mathrm{cr}(\Gamma[v \mapsto f]) + \Delta_{f,g}$. This proves the claim.

Let f be the face of $A(\Gamma, v)$ containing v. In order to find a face $f^\star$ with the minimum number of crossings $\mathrm{cr}(\Gamma[v \mapsto f^\star])$, we determine the number of crossing $\mathrm{cr}(\Gamma[v \mapsto g])$ for every face g in the arrangement $A(\Gamma, v)$. First, we compute the differences $\Delta_{f,g}$ for all adjacent faces. According to Claim 2 this requires $O((kn+m)^2)$ time.

In time linear in the size of $A(\Gamma, v)$ the values $\Gamma[v \mapsto g]$ can be accumulated as described in Claim 3 with a breadth-first search in the dual of $A(\Gamma, v)$ starting at the dual vertex of f. Note that in order to determine the face $f^\star$, the term $\mathrm{cr}(\Gamma)$ can be omitted from the statement of Claim 3, and thus, does not need to be computed. According to Claim 1, the size of $A(\Gamma, v)$ is in $O((kn+m)^2)$ and the arrangement can be computed in $O((kn+m)^2 \log(kn+m))$ time. This concludes the proof. □

4.4 Evaluation

In the following evaluation we consider three approaches (i) our geometric heuristic to minimize the number of crossings, (ii) commonly used algorithms to compute straight-line drawings of arbitrary graphs, i.e. energy-based algorithms, and (iii) an approach to minimize the number of crossings in topological drawings.

We use synthetic and real-world instances to evaluate the performance of the algorithms. Section 4.4.1 contains a brief description of our benchmark instances. The evaluation is based on descriptive statistics and the statistical test described in Section 3. Our evaluation is structured as follows. First, we identify a representative for each type of heuristic, i.e., in Section 4.4.3 we consider energy-based layouts, in Section 4.4.4, Section 4.4.5 and Section 4.4.6 we consider several configurations of the vertex-movement, vertex-insertion and edge-insertion approach, respectively.

Starting from Section 4.4.7 we compare the representatives to each other. In particular in Section 4.4.7 we focus on the vertex-movement, vertex-insertion and the edge-insertion approach. Section 4.4.8 compares stress minimization [GKN05], i.e., the representative of the energy-based layouts, to our heuristics. In Section 4.4.9 we compare our heuristics to a topological crossing minimization approach. We conclude the evaluation with an analysis of the running time in Section 4.4.10.

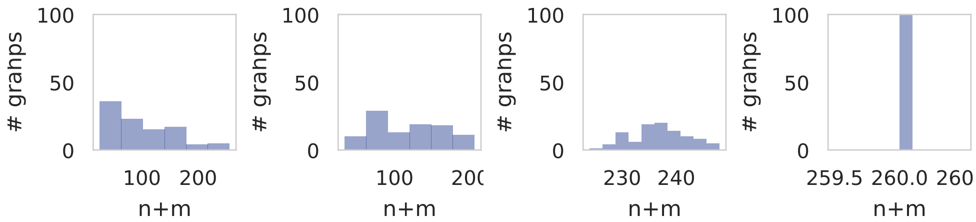
The drawings in Figure 4.6 give a first impression of the effectiveness of our algorithm compared to stress minimization. Figure 4.6a and Figure 4.6c are obtained by one of our heuristics with additional runs of PrEd [Ber00] in order to optimize the aesthetics of the drawing. The remaining two drawings are computed by stress minimization.

All experiments were conducted on a single core of an Intel Xeon(tm) E5-2670 processor clocked at 2.6 GHz. The server is equipped with 64 GB RAM. All algorithms were compiled with g++ version 7.3.1 with optimization mode `-O3`. The operation system was openSUSE Leap 15.0. For geometric operations we rely on CGAL [The17] (v4.10) and GMP[1] to represent coordinates. The usage CGAL and GMP allows us to evaluate our heuristics without dealing with geometric edge cases. We use snapshot 2017-07-23 of OGDF.

4.4.1 Benchmark Instances

We evaluated our algorithms on four classes of graphs, either purely synthetic or with a structure resembling real-world data. The classes North and Rome (AT&T)[2] are the non-planar subsets of the corresponding well known benchmark sets, respectively. The Triangulation+X dataset contains maximal planar graphs with 64 vertices (generated using [BM07]) and ten additional random edges. Note that 64 is the maximal number of vertices the generator of Brinkmann et al. can handle. The Community graphs are generated with the LFR-Generator [LFF08] implemented in NetworKit [SSM16]. They resemble social networks with a community structure. Note that the term *community structure* is not formally defined, i.e., the set of all Community graphs is

[1] `gmplib.org`
[2] http://graphdrawing.org/data.html



(a) North **(b)** Rome **(c)** Community **(d)** Triangulation+X

Figure 4.7: Distribution of the number of vertices plus number of edges for each dataset.

the set of graphs that can be generated with the LFR-generator. Thus, the statistical analysis of the Community graphs in section is with respect to the graph distribution of the LFR-generator. For each of the remaining classes, we selected 100 graphs uniformly at random. Figure 4.7 shows the size distribution of these graphs.

For each graph G we generated a *random drawing* on an $m \times m$ integer grid, i.e., the x- and y-coordinates of each vertex is an integer between 0 and m chosen uniformly at random, where m is the number of edges of G. In case that the drawing contains three collinear vertices, we assign a new random position to one of the three vertices. We repeat this process until all vertices are in general position. The resulting drawing is then used as input for all evaluated algorithms.

4.4.2 Framework for the Evaluation

We use the descriptive and inferential statistical tools introduced in Section 3. Moreover, we show for each algorithm the distribution of the number of crossings in form of a *swarm plot*, compare for example Figure 4.8 In order to keep the extend of the evaluation at a reasonable level, we decided to not distinguish between the four graph classes in Section 4.4.3 to Section 4.4.6. Each of the benchmark sets are drawn uniformly at random from their graph class. Since this is not true anymore for the union of the benchmark set, an inferential test is not meaningful in Section 4.4.3 to Section 4.4.6. Therefore, in this case we provide only descriptive measures, including the advantages. The evaluation in Section 4.4.6 considers the four benchmark sets independently. Therefore, we are able to draw conclusions that are significant at significance level of $\alpha = 0.05$. In particular, we formulate and evaluate hypotheses using the model described in Section 3.2.

4.4.3 Energy-Based Layouts

In this section we evaluate the energy-based layouts implemented in the Open Graph Drawing Framework (OGDF), compare Table 4.1, with respect to the rectilinear

Table 4.1: Energy-based graph drawing algorithms implemented in OGDF.

Name	OGDF	Ref.
DH	OGDF::DavidsonHarel	[DH96]
FMMM	OGDF::FMMMLayout	[HJ05]
FR	OGDF::SpringEmbedderFR	[FR91]
GEM	OGDF::GEMLayout	[FLM95]
KK	OGDF::SpringEmbedderKK	[KK89]
PMDS	OGDF::PivotMDS	[BP07]
Stress	OGDF::StressMinimization	[GKN05]

Table 4.2: Descriptive statistics of the number of crossings.

Algorithm	Mean	Min	.25-Percentile	Median	.75-Percentile	Max
DH	651.59	6	248.50	570.0	993.25	2210
FMMM	156.58	1	35.00	89.0	289.00	1369
FR	163.75	2	46.75	109.0	281.25	1115
GEM	153.43	1	34.25	94.0	259.75	1174
KK	202.20	1	35.75	86.0	327.00	2503
PMDS	198.84	1	37.75	99.0	307.25	2449
Stress	155.78	1	32.75	82.5	288.75	1220

crossing number. Some drawings computed by FMMM, KK and PMDS are *not valid*, i.e., distinct vertices have the same coordinates or a vertex lies in the interior of an edge. We resolve this issue by iteratively perturbing vertices that lie on the interior of an edge.

According to Table 4.2 drawings computed by DH have a considerably higher number of crossings than drawings computed by GEM, FR or Stress. The table indicates that FR computes drawings with a slightly higher number of crossings compared to Stress and GEM. A comparison of Stress and GEM is not conclusive, e.g., Stress has larger mean but a smaller median. Observe that FMMM computes drawings with only small number of crossings more than Stress. Note that the objective function of DH is explicitly configured to minimize the number of crossings. The remaining algorithms do not have explicit mechanisms to reduce the crossings.

Each point in the plot in Figure 4.8 corresponds to the number of crossings of one drawing computed by the algorithm indicated by the color. The measurements are categorized by the respective graph class. We removed *outliers* from the plot, i.e., the plot shows all measurements that differ by at most three times the standard deviation from the mean of the respective datasets. The plot confirms our observation that DH computes drawings with the highest number of crossings. For the remaining algorithms, the plot does not show a clear preference. Comparing the graph classes to

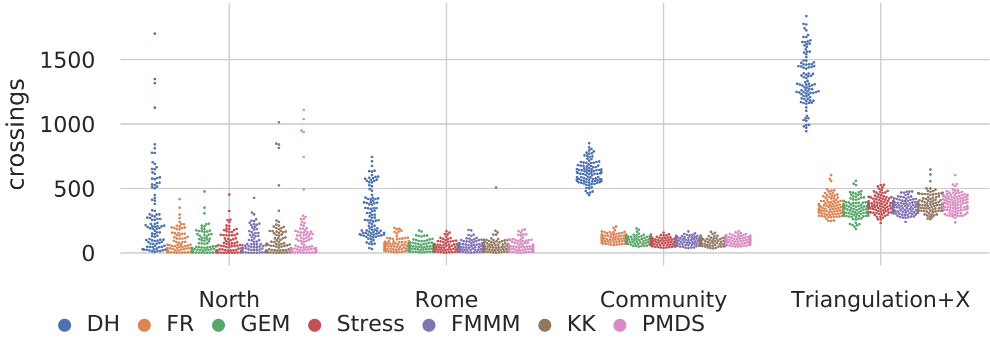


Figure 4.8: Comparison of drawings obtained from algorithms implemented in OGDF. The number of crossings of a drawing for each graph in the class indicated on the x-axis clustered by the algorithms. Outliers have been removed.

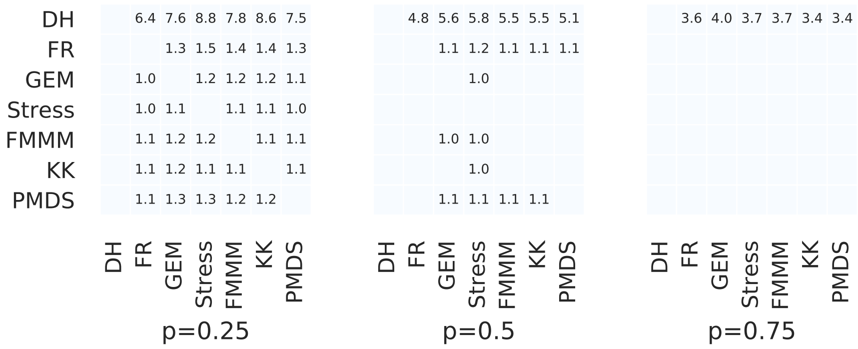


Figure 4.9: Advantages of pairs of algorithms.

each other, the plot indicates that the drawings of graphs in the class Triangulation+X computed by energy-based algorithms tend to have a larger number of crossings in comparison to the remaining classes.

The observations drawn from Table 4.2 and Figure 4.8 neglect the fact that the algorithms compute drawings of the same graphs, i.e., we are able to directly compare the number of crossings of the drawings. The concept of advantages of one set of drawings over another set of drawings introduced in Section 3 uses the mapping between the drawings to compare the drawings. Note that the following advantages are not significant. Figure 4.9 shows the advantages of the algorithms on the x-axis over the algorithms on the y-axis. For example, Stress has an advantage of 3.7 over DH, for $p = 0.75$, i.e., the number of crossings of at least 75% of the drawings computed by DH are larger by a factor of 3.7 than in the corresponding drawings computed by Stress. For $p = 0.5$, we observe that Stress has an advantage over all algorithms. The advantages in between 1.0 and 1.2. We conclude that STRESS computes drawings with a slightly smaller number of crossings in comparison to the other energy-based layouts. Thus, in the following we use STRESS as a representative for the class of energy-based algorithms.

4.4.4 Vertex Movement

For the vertex movement approach described in Section 4.2.1 we are free to choose a vertex order. In this section, we evaluate how the choice of the vertex order affects the number of crossings of the final drawings.

In Section 4.2.1 we introduced three possibilities to count the number of crossings for a vertex v of G. Moreover, we can decide to order the vertices in ascending or in descending order. Table 4.3 lists all configurations of the vertex movement approach that we evaluate. It contains additionally a random permutation of the vertex set.

Table 4.3: Different possibilities to order the vertices.

Name	Counting for $v \in V$	Order
ASC_LOG	$\mathrm{cr}_{\mathrm{LOG}}(v)$	ascending
ASC_SUM	$\mathrm{cr}_{\mathrm{SUM}}(v)$	ascending
ASC_SQ	$\mathrm{cr}_{\mathrm{SQ}}(v)$	ascending
DESC_LOG	$\mathrm{cr}_{\mathrm{LOG}}(v)$	descending
DESC_SUM	$\mathrm{cr}_{\mathrm{SUM}}(v)$	descending
DESC_SQ	$\mathrm{cr}_{\mathrm{SQ}}(v)$	descending
RND		random

As in the evaluation of the energy-based layouts we use the descriptive statistics in Table 4.4, the plot in Figure 4.10 and the advantages (Figure 4.11) to compare the configurations of the vertex-movement approach to each other. The statistics in

Table 4.4: Descriptive statistics of the number of crossings obtained by the *vertex-movement* approach with different vertex orders.

Algorithm	Mean	Min	.25-Percentile	Median	.75-Percentile	Max
ASC_LOG	282.35	1	44.00	214.5	495.50	1147
ASC_SUM	303.83	1	47.00	233.0	504.50	1303
ASC_SQ	310.28	1	46.75	246.5	510.25	1184
DESC_LOG	182.28	1	32.75	167.5	267.25	1074
DESC_SUM	176.24	1	29.00	157.5	258.75	970
DESC_SQ	174.46	1	30.75	157.0	266.00	910
RND	238.54	1	35.75	185.5	387.00	1070

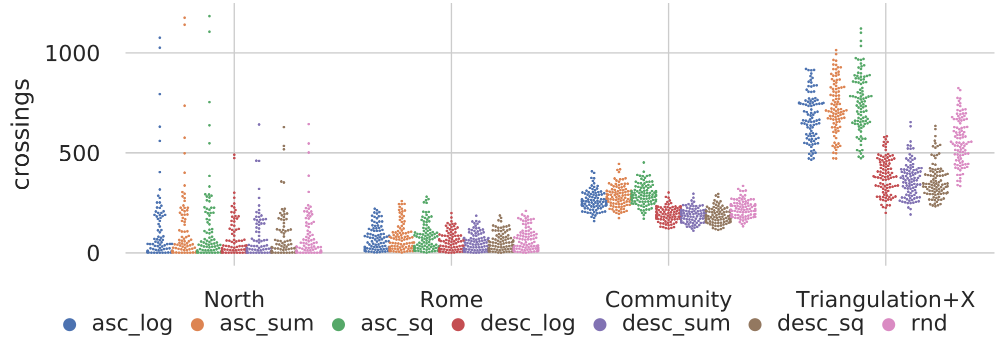


Figure 4.10: Comparison of drawings obtained from different configurations of the vertex movement approach. The number of crossings of a drawing for each graph in the class indicated on the x-axis clustered by the configurations. Outliers have been removed.

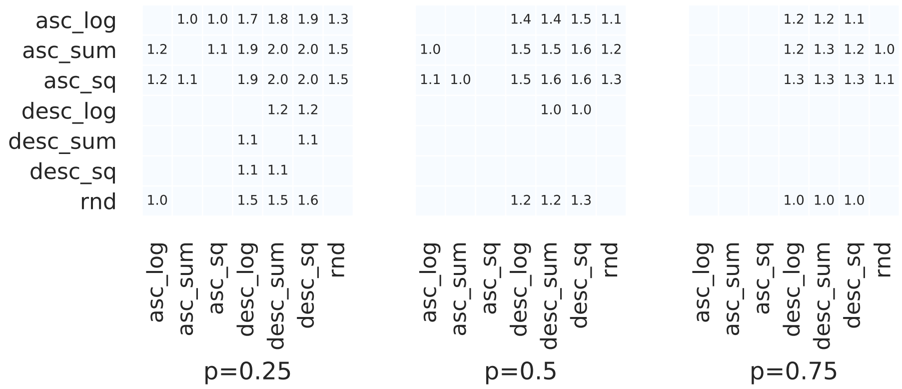


Figure 4.11: Advantages of pairs of configurations of the vertex movement approach.

Table 4.4 and the plot in Figure 4.10 indicate that configurations that move the vertices in ascending order (Asc_★) compute drawings with a considerably higher number of crossings compared to configurations using a descending order (Desc_★) or the random order (Rnd).

The plots of the advantages in Figure 4.11 confirm this observation for $p = 0.75$. For plots do not show a clear preference for either Desc_Sum configuration or the Desc_Sq configuration. For $p = 0.5$ both have an advantage of 1.0 over Desc_Log and for the $p = 0.25$, the corresponding advantage is 1.2.

In order to reduce the complexity of the rest of the evaluation, we choose a single configuration of the vertex movement approach. Therefore, we consider the average number of crossings as a tie breaker. Since the Desc_Sq computes the smallest number of crossings with respect to this statistic, we use this configuration as the representative for the vertex-movement approach.

Table 4.5: Descriptive statistics of the number of crossings obtained by the *vertex-insertion* approach with different vertex orders.

Algorithm	Mean	Min	.25-Percentile	Median	.75-Percentile	Max
Asc_Log	126.18	1	37.00	138.5	185.25	1217
Asc_Sum	131.16	1	34.00	142.0	188.00	1187
Asc_Sq	140.57	1	37.00	155.5	204.00	1316
Desc_Log	406.32	1	76.75	310.0	778.50	1905
Desc_Sum	386.56	1	72.75	276.5	739.75	1712
Desc_Sq	360.68	1	62.00	252.0	680.25	1652
Rnd	237.01	1	55.50	227.5	369.25	1080

4.4.5 Vertex Insertion

Similar to the vertex-movement approach, the order in which we remove and insert vertices in the vertex-insertion approach (Section 4.2.2), can affect the number of crossings of the final drawing. In this section, we evaluate the vertex-insertion approach with different vertex orders (see Table 4.3). Note that in case of an ascending order (Asc_⋆) the vertices are removed in this order and inserted in the reversed (descending) order. Vice versa, in an descending order (Desc_⋆) the vertices are removed in descending order and reinserted in ascending order. Preliminary experiments indicated that reinserting the vertices in same order instead of the reversed order yields a larger number crossings. In order to reduce the complexity of the evaluation, we decided to omit these configurations.

The descriptive statistics in Table 4.5 and the plot in Figure 4.12 show that the descending vertex orders yield drawings with a considerably higher number of crossings compared to the ascending orders. The statistics indicate that the vertex insertion approach with the Asc_Log order computes drawings with the smallest number of crossings. The advantages in Figure 4.13 confirm this observation. For $p = 0.75$, Asc_Log the advantage Δ over any other configuration C is higher or equal to the corresponding advantage of Asc_Sum and Asc_Sq over C. Moreover, for $p = 0.25$ the Asc_Log order has an advantage of 1.2 and 1.3 over Asc_Sum and Asc_Sq, respectively. On the other hand, Asc_Sum and Asc_Sq each have only an advantage of 1.1 over Asc_Log. Hence, for the following evaluations we consider the vertex-insertion approach with the Asc_Log order.

4.4.6 Edge Insertion

The edge insertion approach as described in Section 4.2.3 has several degrees of freedom: (i) the computation of the maximal planar subgraph, (ii) the initial drawing of the maximal planar subgraph, (iii) the order in which the edges are reinserted,

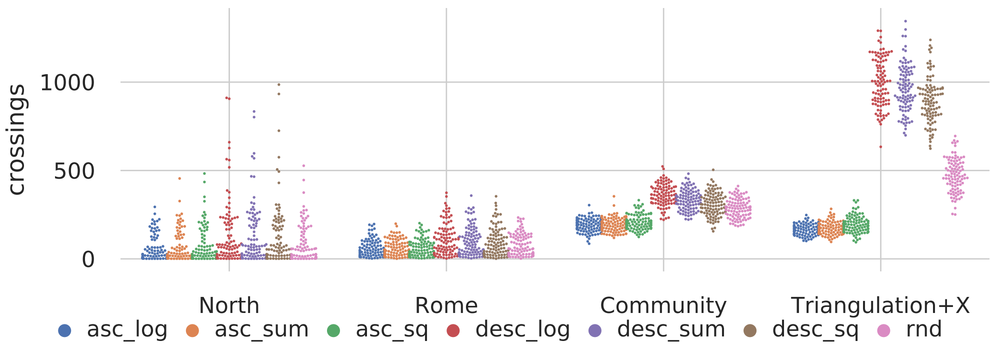


Figure 4.12: Comparison of drawings obtained from different configurations of the vertex insertion approach. The number of crossings of a drawing for each graph in the class indicated on the x-axis clustered by the configurations. Outliers have been removed.

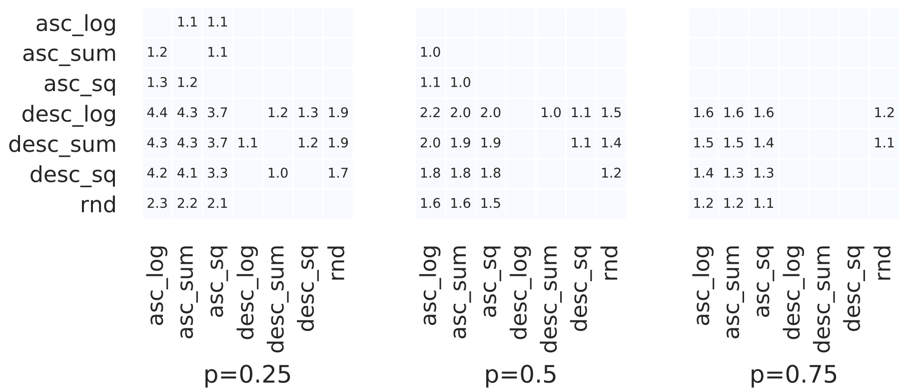


Figure 4.13: Advantages of pairs of configurations of the vertex insertion approach.

and (iv) the order in which the vertices are moved after each edge insertion. We use the PlanarSubgraphFast algorithm [JLM98] implemented in OGDF in order to compute a large planar subgraph and the PlanarStraightLayout method in order to compute an initial drawing of the planar subgraph. In both cases we use the default configuration of OGDF. We reinsert the edges in the order they are returned by the PlanarSubgraphFast routine. The configuration EP only moves the endpoints of an edge. The configuration EI moves in addition to the endpoints the crossed neighborhood of the newly inserted edge e, i.e., vertices that are incident to an edge that crosses e. In Section 4.4.4 we selected the Desc_Sq order as a representative for the vertex-movement approach. Hence, we use this vertex order to move the crossed neighborhood of an edge.

Since EI moves a superset of the vertices of EP, we expect that EI further reduces the number of crossings, compared to EP. Table 4.7, the plots in Figure 4.14 and Figure 4.15 confirm this observation.

In comparison to the conference version [Rad+18] of the chapter, we reimplemented the geometric operation of moving a single vertex and the heuristics (VM, VI, EI, EP). In the experiments on the old code base, we observed that the edge insertion heuristic with the additional movements of subgraphs introduced a significant number of new crossings. Since moving entire subgraphs did not seem promising, we decided to not reimplemented this particular heuristic.

4.4.7 Comparison of our Heuristics.

In the following we compare our heuristics, i.e. VM, VI, EP and EI, to each other; refer to Table 4.6. For the comparison of the heuristics to Stress and Tpl refer to Section 4.4.8 and Section 4.4.9, respectively. Table 4.7 suggests that EI computes drawings with fewer crossings than EP, EP fewer than VI and VM. Recall that a point in Figure 4.14 corresponds to the number of crossings of one drawing computed by the algorithm indicated by the color. The plot confirms that the edge-insertion approaches compute drawings with fewer crossings than VI and VM. Moreover, VI computes drawings of the Triangulation+X graphs with fewer crossings then VM. For $p = 0.75$, we observe that EI and EP compute drawings with considerably fewer crossings than VI and VM; refer to Figure 4.15.

We now consider the graph classes independently. This enables us to draw statements at a significance level of $\alpha = 0.05$. The hypothesis are generated in two phases as described in Chapter 3, i.e., we determine a maximum advantage Δ for each pair of algorithms and each $p \in [0.25, 0.5, 0.75]$ on a test set that contain 50% of each benchmark set. On the remaining 50% of graphs we check whether there is a significant advantage of $0.75 \cdot \Delta$ for the respective algorithms. Each graph is assigned to either the test or the verification set uniformly at random. The plots for the North, Rome, Community and Triangulation+X graphs are given in Figure 4.16 Figure 4.17, Figure 4.18 and Figure 4.19, respectively. If the background color of a cell is blue, it indicates the advantage in this cell is significant. Otherwise, we were not able to reject the Null-Hypothesis at a significance level of $\alpha = 0.05$.

For the Triangulation+X graphs only EI has a significant advantage over both VI and VM, for $p = 0.75$. Thus, for a Triangulation+X graph selected uniformly at random it is unlikely that the probability that EI has an advantage over VI (VM) is less than $p = 0.75$. For $p = 0.5$, EP has a significant advantage of 1.8 and 1.0 over VM and VI, respectively.

Only for Triangulation+X graphs and the Community graphs, EI has a significant advantage of 1.0 over EP, for $p = 0.75$. For $p = 0.5$, EI has an advantage of 1.0, on the

Table 4.6: Configuration of the reference algorithms.

	Algorithm	Vertex Order
VM	Vertex Movement	Desc_Sq
VI	Vertex Insertion	Asc_Log
EP	Edge Insertion	Endpoints
EI	Edge Insertion	EP + Crossed Neighbors (Desc_Sq)
Stress	ogdf::StressMinimization	
Tpl	ogdf::SubgraphPlanarizer	

Table 4.7: Descriptive statistics of the number of crossings of drawings computed by the final configuration of the heuristics, Stress and Tpl.

Algorithm	Mean	Min	.25-Percentile	Median	.75-Percentile	Max
Tpl	43.30	1	7.00	29.0	66.25	610
EI	55.43	1	9.00	41.0	87.25	601
EP	69.41	1	9.00	49.0	107.75	630
VI	126.18	1	37.00	138.5	185.25	1217
VM	174.46	1	30.75	157.0	266.00	910
Stress	155.51	1	32.75	82.5	288.75	1220

Rome graphs. Note that in contrast to $p = 0.25$ this advantage is not significant. For the North graphs and $p = 0.25$, EI has a significant advantage of 1.0 over EP.

Comparing VI and VM on the union of all benchmark sets, VI has an advantage of 1.0 over VM for $p = 0.5$; see Figure 4.15. For $p = 0.25$, VI has an advantage of 1.6 over VM and VM has an advantage of 1.2 over VI. Considering the graph classes independently, we see that on the North, Rome and Community graphs, VM has a small advantage over VI again for $p = 0.25$. For the North and Community graphs these advantages are significant. On the Triangulation+X graphs VI computes drawings with considerably fewer crossings than VM, i.e., for $p = 0.75$ VI has an significant advantage of 1.4 over VM.

Overall, we conclude that the edge-insertion approach (EI and EP) computes drawings with significantly fewer crossings than its competitors. It For $p = 0.25$ this advantage increases to 1.2. depends on the graph class, whether the additional movement of vertices (EI) significantly decreases the number of crossings compared to EP.

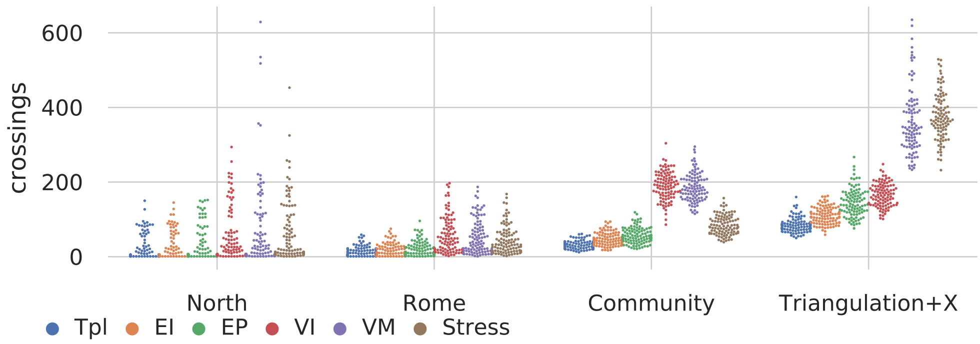


Figure 4.14: Comparison of the final configurations of each heuristic, Stress and Tpl. The number of crossings of a drawing for each graph in the class indicated on the x-axis clustered by the heuristic. Outliers have been removed.

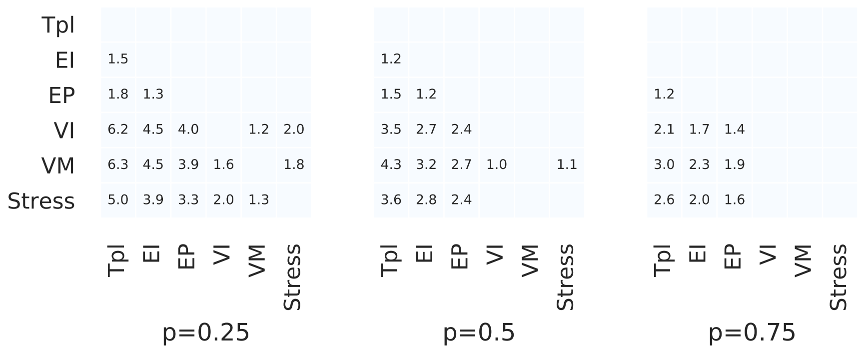


Figure 4.15: Advantages of pairs of the final configurations of the heuristics, Stress and Tpl.

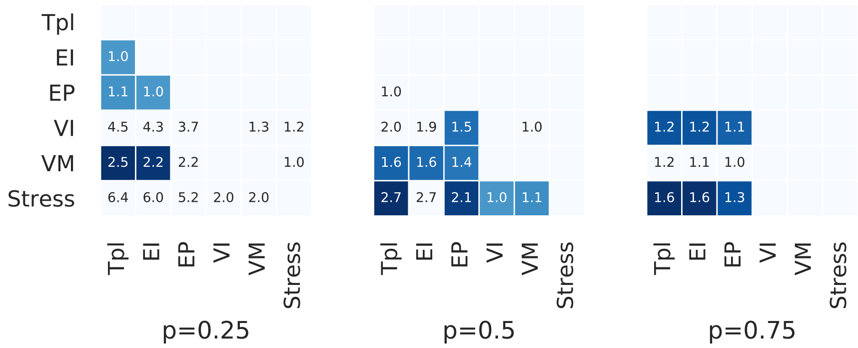


Figure 4.16: North

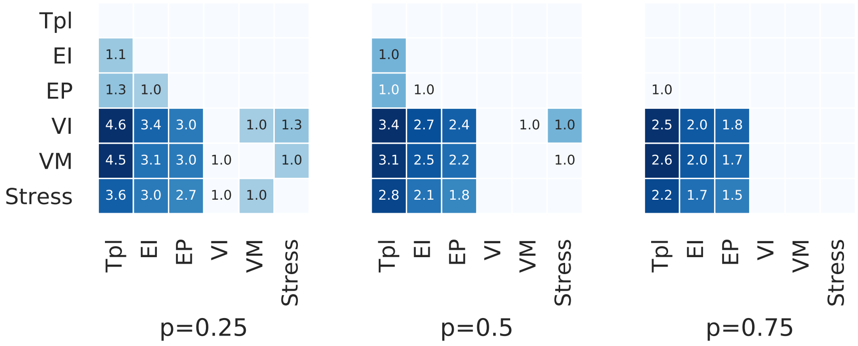


Figure 4.17: ROME

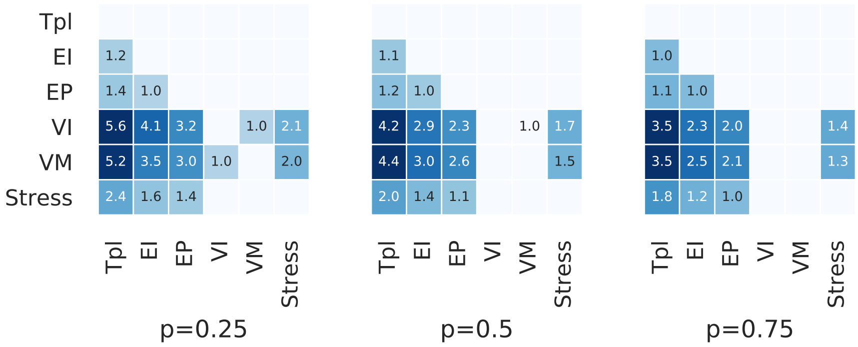


Figure 4.18: COMMUNITY

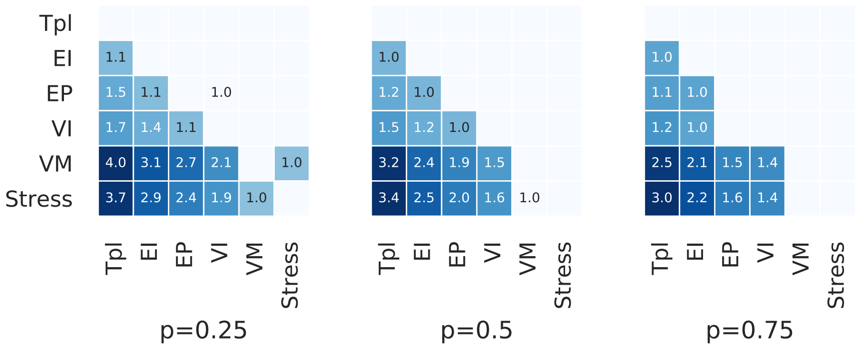


Figure 4.19: TRIANGULATION+X

4.4.8 Comparison to Stress.

We compare the drawings computed by our heuristics with drawings computed by stress minimization (Stress), i.e., to an algorithm commonly used to compute straight-line drawings of general graphs. In Section 4.4.3 showed that this algorithm computes drawings with fewer crossings than other energy-based heuristics implemented in OGDF. We configured Stress to stop after convergence, thus we can not expect Stress to compute drawings with a smaller number of crossings if we increase the computing time.

Table 4.7 suggests that Stress computes drawings with at least a factor two more crossings than EI and EP. A comparison between Stress and VI is inconclusive. On average VI computes drawings with a smaller number of crossings; on the other hand, Stress has a smaller median value.

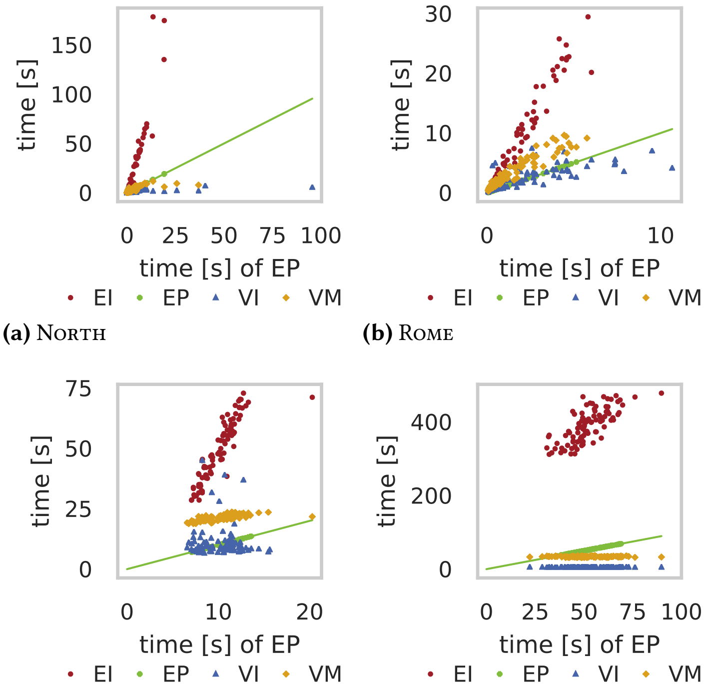
In addition to the above observations, Figure 4.14 shows that on a large subset of the Triangulation+X graphs Stress computes drawings with a considerably larger amount of crossings than EI, EP and VI. On the Community graphs Stress achieves a smaller number of crossings than VI and VM. For the remaining graph classes the plot provides no clear distinction between VI, VM and Stress. Although Table 4.7 and Figure 4.14 do not provide a conclusive distinction between Stress and VM, Figure 4.15 shows that Stress has an advantage of 1.1 over VM, for $p = 0.5$.

The advantages in Figure 4.15 show that Stress computes drawings with a factor of 2.0 and 1.6 more crossings than EI and EP, respectively, for $p = 0.75$. Further, considering only the Community graphs (Figure 4.18), EI has a significant advantage of 1.2 over Stress, for $p = 0.75$. For the Triangulation+X graphs, the advantage increases to 2.2. We conclude that the edge-insertion approach computes drawings with significantly fewer crossings than Stress.

4.4.9 Comparison to Tpl.

We investigate how close the number of crossings in drawings computed by EI are to the number of crossings in topological drawings. Note that Tpl as well as EI start from a large planar subgraph and iteratively insert the remaining edges. The drawings obtained by Tpl are not necessarily realizable as straight-line drawings with the same number of crossings.

Table 4.7 shows that the maximum number of crossings computed by EI is smaller than the corresponding number computed by Tpl. The Tpl approach iteratively inserts edges into a planar graph. After each edge insertion the crossings are replaced by degree four vertices. This fixes the crossings for future edge insertions. Our edge insertion approach (EI) at least moves the vertices v incident to a new edge e. Since the vertex movement minimizes the number of crossings of all edges incident to v, it is possible that two edges that cross in Γ do not cross in Γ_{+e}. Apparently, this flexibility



(a) NORTH (b) ROME (c) COMMUNITY (d) TRIANGULATION+X

Figure 4.20: The running time of each algorithm as a function of the running time of EP, i.e., each data point (t_x, t_y) corresponds to graph G and an algorithm $\mathcal{A}$, where t_x and t_y is the running time of EP and $\mathcal{A}$ on G, respectively. We removed outliers to increase readability.

helps in some cases to find drawings with fewer crossings compared to TPL. Indeed there are 60 out of 400 instances in which the number of crossings computed by EI is smaller or equal to the number of crossings computed by TPL. On 35 instances EI achieves a strictly smaller number of crossings than TPL.

For at least 75% of graphs TPL has an advantage of 1.1 over EP, see Figure 4.15. For the same number of graphs, TPL does not have an advantage over EI. On the other hand, there is a subset containing at least 25% of graphs such that TPL has an advantage of 1.5 over EI, and 1.8 over EP. Considering the COMMUNITY and the TRIANGULATION+X graphs, TPL has a significant advantage over all other algorithms for $p = 0.75$, but the advantage over EI is only 1.0.

Table 4.8: Descriptive statistics of the running time in seconds per graph class. Percentile is abbreviated with Perc.

Algorithm	Mean	Min	.25-Perc.	Median	.75-Perc.	Max
North						
Tpl	0.90	< 0.01	< 0.01	0.01	0.16	42.07
EI	35.77	0.04	0.46	2.61	30.64	934.79
EP	4.37	0.01	0.10	0.57	4.60	95.53
VI	1.49	0.02	0.17	0.69	2.09	9.23
VM	3.86	0.04	0.32	1.48	5.87	29.60
Rome						
Tpl	0.06	< 0.01	0.01	0.03	0.08	0.55
EI	10.86	0.20	1.14	4.99	15.81	61.11
EP	2.12	0.07	0.32	1.11	2.94	10.67
VI	5.62	0.09	0.66	1.85	3.24	178.40
VM	4.20	0.27	1.12	2.70	6.19	15.31
Community						
Tpl	0.24	0.04	0.13	0.20	0.29	0.90
EI	50.81	24.66	38.80	50.41	61.11	88.73
EP	10.28	6.58	8.26	10.43	11.64	20.29
VI	27.40	7.05	8.60	10.25	17.53	514.47
VM	21.24	17.34	20.10	21.27	22.26	25.61
Tri. + X						
Tpl	0.67	0.10	0.40	0.57	0.83	2.79
EI	391.40	200.23	348.75	393.31	428.81	566.14
EP	52.20	22.23	45.28	50.56	60.47	89.74
VI	5.56	4.98	5.39	5.53	5.71	6.52
VM	34.17	31.28	33.06	34.06	34.99	50.46

4.4.10 Running Time

In this section we analyze the running time of our algorithms. We abstain from a comparison to Stress, since Stress is very well engineered and requires at most 10^{-2} seconds per instance on our graphs.

We compare the remaining algorithms listed in Table 4.6. Table 4.8 shows several statistics of the running time grouped by graph class. Figure 4.20 shows the running time of each instance for all graph classes. Since the running time of Tpl is less than one second for most instances, compare Table 4.8, we omit these measurements in Figure 4.20 to increase readability. A data point p_G below the green diagonal indicates that the algorithm that corresponds to p_G uses less time to finish on the graph than EP. For example, Figure 4.20d shows that there are many instances where VM and VI consume less time than EP. On the other hand, on every Triangulation+X instance, the running time of EI is considerably higher. Note that on the Triangulation+X graphs, EI only has a small advantage over VI; compare Figure 4.19. On the other hand, VI is significantly faster on this graph class.

The observation that EI has the longest running time, is true for all graph classes. Recall that EI moves a superset of vertices compared to EP. Thus, this observation is expected. Moreover, the figures show that the edge-insertion approach that only moves endpoints of an edge (EP) and VI profit from the incremental growth of the drawing, whereas the vertex-movement approach has to deal with the entire graph in each iteration.

4.5 Conclusion

In this chapter we introduced several heuristics that are based on moving a vertex to its crossing minimal position. This position can be computed in $O((kn+m)^2 \log(kn+m))$ time. Our evaluation in Section 4.4 shows that the approach yields drawings with a smaller number of crossings in comparison to the well-established stress minimization algorithm.

The edge-insertion approach in combination with the crossed neighborhood strategy computes drawings with the smallest number of crossings. We compared our heuristic to an approach computing topological drawings with a small number of crossings. Our experimental evaluation showed that there is only a relatively small difference between the number of crossings. Especially, we could show that we are able to match the number of crossings in about 15% of our instances. In Chapter 5 we engineer the approach to cope with larger instances.

5 Scaleable Crossing Minimization

We consider the minimization of edge-crossings in geometric drawings of graphs $G = (V, E)$, i.e., in drawings where each edge is depicted as a line segment. The respective decision problem is $\mathcal{NP}$-hard [Bie91]. Crossing-minimization, in general, is a popular theoretical research topic; see Vrt'o [Vrt14]. In contrast to theory and the topological setting, the geometric setting did not receive a lot of attention in practice. The approach introduced in Chapter 4 is limited to the crossing-minimization in geometric graphs with less than 200 edges. The described heuristics base on the primitive operation of moving a single vertex v to its *crossing-minimal position*, i.e., a position in $\mathbb{R}^2$ that minimizes the number of crossings on edges incident to v.

In this chapter, we introduce a technique to speed-up the computation by a factor of 20. This is necessary but not sufficient to cope with graphs with a few thousand edges. In order to handle larger graphs, we drop the condition that each vertex v has to be moved to its crossing-minimal position and compute a position that is only optimal with respect to a small random subset of the edges. In our theoretical contribution, we consider drawings that contain for each edge $uv \in E$ and each position $p \in \mathbb{R}^2$ for v $o(|E|)$ crossings. In this case, we prove that with a random subset of the edges of size $\Theta(k \log k)$ the *co-crossing* number of a degree-k vertex v, i.e., the number of edge pairs $uv \in E, e \in E$ that do *not* cross, can be approximated by an arbitrary but fixed factor δ with high probability. In our experimental evaluation, we show that the randomized approach reduces the number of crossings in graphs with up to 12 000 edges considerably. The evaluation suggests that depending on the degree-distribution different strategies result in the fewest number of crossings.

This chapter is based on joint work with Ignaz Rutter [RR19].

5.1 Introduction

The minimization of crossings in geometric drawings of graphs is a fundamental graph drawing problem. In general the problem is $\mathcal{NP}$-hard [Bie91, GJ83] and has been studied from numerous theoretical perspectives; see Vrt'o [Vrt14]. Until recently only the topological setting, where edges are drawn as topological curves, has been considered in practice [Buc+13, CH16, GMW05]. In Chapter 4 we describe geometric heuristics that compute straight-line drawings of graphs with significantly fewer crossings compared to common energy-based layouts. One of the heuristics is the *vertex-movement approach* that iteratively moves a single vertex v to its *crossing-minimal position*, i.e., a position $p^\star$ so that crossings of edges incident to v are minimized. Unfortunately, the worst-case running time to compute this position is super-quadratic in the size of the graph as the following theorem states.

Theorem 5.1 (Theorem 4.2 in Chapter 4)**.** *The crossing-minimal position of a degree-k vertex v with respect to a straight-line drawing Γ of a graph $G = (V, E)$ can be computed in $O((kn + m)^2 \log(kn + m))$ time, where $n = |V|, m = |E|$.*

This is not only a theoretical upper bound on the running time but is also a limitation that has been observed in practice. The implementation we used previously requires considerable time to compute drawings with few crossings. For this reason we were only able evaluate our approach on graphs with at most 200 edges. For example, on a class of graphs that have 64 vertices and 196 edges our implementation already required on average about 35 seconds to compute a drawing with few crossings.

Energy-based methods are common and well engineered tools to draw any graph. For an overview we refer to [Kob13]. For example, the aim of *Stress Majorization* (or simply Stress) is to compute a drawing such that the Euclidean distance of each two vertices corresponds to their graph-theoretical distance [GKN05]. The algorithm has been engineered to handle graphs with up to 10^6 vertices and $3 \cdot 10^6$ edges [MNS18]. Kobourov [Kob13] claimed that Stress tends to crossing-free drawing for planar graphs. In the experimental evaluation in Chapter 4 we demonstrated for varied set of graph classes that we are able compute drawings with significantly less crossings than drawings computed by Stress.

Fabila-Monroy and López [FL14] introduced a randomized algorithm to compute a drawing of K_n with a small number of crossings. Many best known upper bounds on the rectilinear crossing number of K_n, for $44 \leq k \leq 99$, are due to this approach [Aic19]. The algorithm iteratively updates a set P of n points, by replacing a random point $p \in P$ by a random point q that is close to p, if q improves the number of crossings. Since the number of crossings of K_n is in $\Theta(n^4)$, the bottleneck of their approach is the running time for counting the number of crossings induced by P. A similar randomized approach has been used to maximize the smallest crossing angle in a straight-line drawing; compare Chapter 7 and [Bek+18]. The approach iteratively moves vertices to the best position within a random point set.

Contribution. The main contribution of this chapter is to engineer the *vertex-movement approach* for the minimization of crossings in geometric drawings described in Chapter 4 to be applicable on graphs with a few thousands vertices and edges.

1. In Section 5.3 we introduce so-called *bloated duals of line arrangements*, a combinatorial technique to construct a dual representation of general line arrangements. In our application this results in an overall speed-up of about a factor of

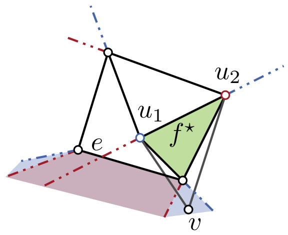


Figure 5.1: The black, blue and red segments show the arrangement $A(\Gamma, v)$ of the black drawing Γ. The blue and red region show the complement of the visibility regions of u_1 and u_2, respectively, and the edge e. The green region is crossing minimal.

20 in comparison to the recent implementation. This speed-up is necessary but not sufficient to handle graphs with a few thousands vertices and edges.

2. In Section 5.4 we demonstrate that taking a small random subset of the edges is sufficient to compute drawings with few crossings. Moreover, in Section 5.4.1 we prove that under certain conditions the randomized approach is an approximation of the co-crossing number of a vertex, with high probability.

3. Based on the insights of the evaluation in Section 5.4.2, we introduce a weighted sampling approach. A comparison to a restrictive approach of sampling points suggests that the degree-distribution of the graph is a good indicator to decide which approach results in fewer crossings.

4. Overall, our experimental evaluation shows that we are now able to handle graphs with 12 000 edges, which are 60 times more than the graphs that have been considered in the evaluation in Chapter 4.

5.2 Preliminaries

We repeat some notation from Chapter 4. Let Γ be a straight-line drawing of a $G = (V, E)$. Denote by $N(v) \subseteq V$ the set of neighbors of v and by $E(v) \subseteq E$ the set of edges incident to v. For a vertex $v \in V$, denote by $\Gamma[v \mapsto p]$ the drawing that is obtained from Γ by moving the vertex v to the point p. We denote the number of crossings in a drawing Γ by $\mathrm{cr}(\Gamma)$, the number of crossings on edges incident to v by $\mathrm{cr}(\Gamma, v)$, and we refer with $\mathrm{cr}(\Gamma, e, f)$ to the number of crossings on two edges e and f in Γ, i.e., $\mathrm{cr}(\Gamma, e, f) \in \{0, 1\}$ if $e \neq f$. For a point u and a segment e, denote by $\mathcal{VR}(u, e)$ the *visibility region of u and e*, i.e., the set of points $p \in \mathbb{R}^2$ such that the segment up and e do not intersect. Moreover, let $\mathcal{BD}(u, e)$ be the boundary of $\mathcal{VR}(u, e)$. Let $A(\Gamma, v)$ be the arrangement over all boundaries $\mathcal{BD}(u, e)$ for each neighbor $u \in N(v)$ of v and each edge $e \in E \setminus E(u)$; see Figure 5.1. The arrangement has the property that two points p and q in a common cell of $A(\Gamma, v)$ induce the same number of crossings for v, i.e., $\mathrm{cr}(\Gamma[v \mapsto p], v) = \mathrm{cr}(\Gamma[v \mapsto q], v)$; see Lemma 4.1. Thus, the computation of a crossing minimal position $p^\star$ reduces to finding a *crossing-minimal region* $f^\star$ in $A(\Gamma, v)$.

For our experiments, we used two different compute servers. Both systems ran with an openSUSE Leap 15.0 operating system. All algorithms were compiled with g++ version 7.3.1 with optimization mode `-O3`. System 1 was used for running time experiments, i.e., for the experiments evaluated in Section 5.3.1 and in Section 5.4.2. System 2 is used for the experiments evaluated in Section 5.4.3.

System 1 Intel Xeon(tm) E5-1630v3 processor clocked at 3.7 GHz, 128 GB RAM.

System 2 Two Intel Xeon(tm) E5-2670 CPU processors clocked at 2.6 GHz, 64 GB RAM.

5.3 Efficient Implementation of the Crossing-Minimal Position

The vertex-movement approach iteratively moves a single vertex to its crossing-minimal position. The running time of the overall algorithm crucially depends on an efficient computation of this operation. Therefore the aim of this section is to provide an efficient implementation of the crossing-minimal position of a vertex. The implementation used for the evaluation in Chapter 4 heavily relies on CGAL [The17], which follows an exact computations paradigm and uses exact number types to, e.g., represent coordinates and intermediate results. This helps to ensure correctness but considerably increases the running time of the algorithms. We introduce an approach to compute the crossing-minimal position that drastically reduces the usage of exact computations.

Computing a crossing-minimal position of a vertex v is equivalent to computing a crossing-minimal region $f^\star$ in the arrangement $A(\Gamma, v)$. The region $f^\star$ of a vertex v can be computed by a breadth-first search in the dual graph $A(\Gamma, v)^\star$. Thus, the time-consuming steps to compute $f^\star$ are to construct the arrangement $A(\Gamma, v)$ and then to build the dual $A(\Gamma, v)^\star$. Instead of computing the dual $A(\Gamma, v)^\star$ we construct a so-called *bloated dual* $A(\Gamma, v)^+$. The advantage of this approach is that it suffices to compute the set of intersecting segments in $A(\Gamma, v)$ to construct $A(\Gamma, v)^+$ and it is not necessary to compute the arrangement $A(\Gamma, v)$ itself, i.e., the exact coordinates of each intersection.

Let S be a set of line segments and let A be the arrangement of S. A *bloated dual of* A is a graph A^+ that has the following properties (compare Figure 5.2a):

(i) each edge e incident to a face f corresponds to a vertex v_e^f in A^+,

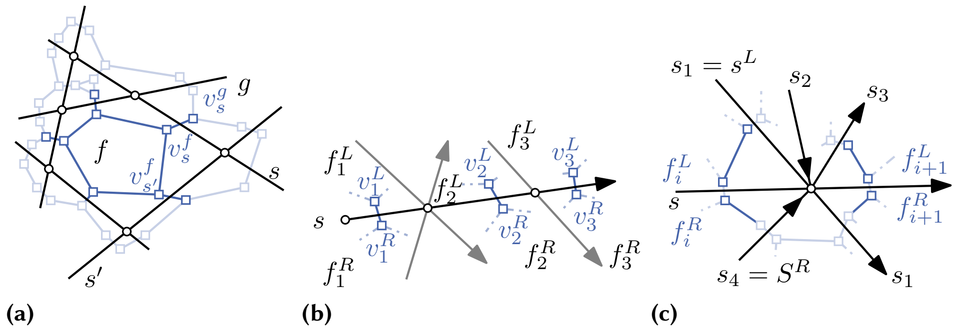


Figure 5.2: (a) Bloated dual A^+ (blue) of an arrangement A (black). Inserting edges dual to a segment s (b) and within a face (c).

(ii) if two distinct segments $s, s' \in S$ of f have a common intersection on the boundary of f, then $v_s^f v_{s'}^f \in E(A^+)$, and

(iii) for two distinct faces f, g sharing a common segment s, there is an edge $v_s^f v_s^g \in E(A^+)$.

Note that given a crossing-minimal face and $v_{s_0}^f$, the geometric representation of f has to be computed in order to compute a crossing-minimal position $p \in f$. Further a vertex $v_{s_0}^f$ belongs to a cycle $v_{s_0}^f, v_{s_1}^f, \ldots v_{s_k}^f$. Then, the geometric representation of the boundary of f can be computed by intersecting the segments s_i and s_{i+1}, where we set $k+1 = 0$. In the following, we will show that it is sufficient to know the order in which the segments in S intersect to construct the bloated dual. Thus, exact number types only have to be used to determine the order of two segments whose intersections with a third segment s have a small distance on s.

We construct the bloated dual of A in two steps. First, we insert all vertices v_s^f, v_s^g and the corresponding edge $v_s^f v_s^g$. In the second step, we insert the remaining edges $v_s^f v_{s'}^f$ within a face f. For a compact description we assume that no intersection point of two segments is an endpoint of a segment. We define the *source of* s and *target of* s to be the lexicographically smallest and largest point on s, respectively. We direct each segment s from its source to its target.

Let $p_1, p_2, \ldots, p_l$ be the intersection points on a segment s in lexicographical order. These intersection points correspond to a set of left faces $f_1^L, f_2^L, \ldots, f_{l+1}^L$ and to a set of right faces $f_1^R, f_2^R, \ldots, f_{l+1}^R$, such that f_i^L and f_i^R share parts of their boundary; see Figure 5.2b. Thus, we can associate a set of vertices $v_i^L, v_i^R, 2 \le i \le l+1$, with s, and add the edges $v_i^L v_i^R$ to A^+. Note that only the order and not the actual coordinates of the points $p_1, \ldots, p_l$ has to be known to insert the edges. Thus, given the set of segments that intersect s, an exact number type is only necessary to determine the order of two segments s_i and s_j whose intersection points p_i and p_j on s have a small distance.

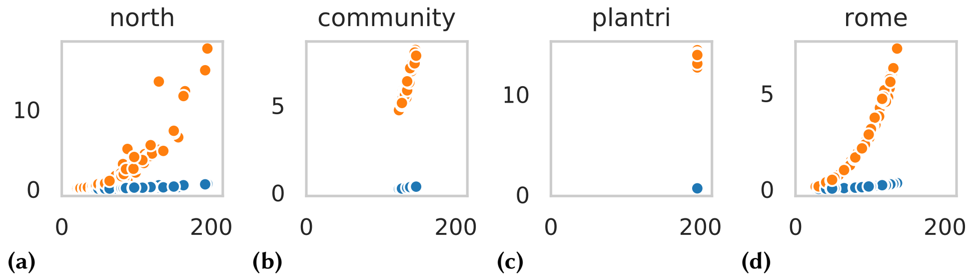


Figure 5.3: Comparing the running time of two approaches (orange Precise, blue Bd) to compute the crossing minimal region. Each point corresponds to a graph G. The x-axis shows the number of edges of G. The y-axis depicts the running time in seconds to compute the crossing minimal regions for all vertices of G.

We now add the remaining edges within a face f. Let $S' = \{s_1, \ldots, s_k\} \subseteq S$ be the set of segments that intersect s in p_i; see Figure 5.2c. The two segments $s^L, s^R \in S'$ that lie on the boundary of f_i^L and f_i^R can be determined as follows. To find the segment s^L, we distinguish two cases. First, assume that there exists a segment $s' \in S'$ whose source is left of s. Observe that if there is a segment s'' whose target is left of s, the segment s'' cannot be the segment s^L. Thus, we assume without loss of generality that all sources of segments in S_s^i are left of s. Then a segment $s' \in S'$ is the segment s^L if and only if the segment s' and each segment $s'' \in S' \setminus \{s'\}$ form a right turn. Now consider the case that there is no segment whose source is left of s. Then a segment s' is s^L if and only if the segment s' and each segment $s'' \in S' \setminus \{s'\}$ form a left turn. The segment s^R can be determined analogously.

Implementation Details. We give some implementation details which allow us to efficiently implement the construction of the bloated dual. We use the index of a vertex to decide whether it is left or right of s, i.e., vertices with an odd index are left of s and vertices with an even index are right of s. The fact that each vertex of A^+ has degree at most 3 allows us to represent A^+ as a single array B of size $3n$, where n is the number of vertices of A^+. The vertices incident to a vertex v_i occupy the cells $B[3i]$, $B[3i+1]$ and $B[3i+2]$. Moreover, each pair of segments in S can be handled independently to construct the bloated dual. This enables a parallelization over the segments in S.

5.3.1 Evaluation of the Running Time

In this section, we compare the running time of the two approaches to compute the crossing-minimal region of a vertex. We refer with Precise to the approach that uses CGAL to compute the crossing minimal region and with Bd to the approach based

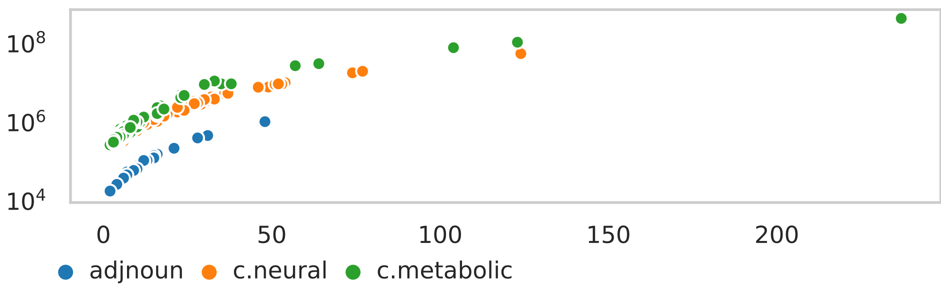


Figure 5.4: The x-axis shows the vertex-degree and the y-axis the number of intersecting edges in the arrangement $A(\Gamma, v)$. The y-axis is in log-scale.

on the bloated dual. In order to compute all intersecting segments, we use a naive implementation of a sweep-line algorithm [BO79]. In this approach all segments within a specific interval are pairwise checked for an intersection. This has the advantage that the computation is independent of the coordinates of the intersection.

The experimental setup is as follows. Given a drawing Γ of a graph G, we are interested in the running time of moving all vertices of a graph to their crossing-minimal positions. Therefore, we measure the running time of computing the crossing-minimal regions of all vertices. In order to guarantee the comparability of the two approaches, we use the same vertex order and only compute the crossing-minimal region but do not update the positions of the vertices. We use the same set of benchmark graphs used in Chapter 4: North[1], Rome[1], graphs that have Community structure, and Triangulations on 64 vertices with an additional 10 random edges. For each graph class, 100 graphs were selected uniformly at random. We use the implementation of Stress [GKN05] provided by Ogdf [Chi+13] (snapshot 2017-07-23) to compute an initial layout of the graphs.

The plots in Figure 5.3 shows the results of the experiments. Each point in the plot corresponds to the running time of computing all crossing-minimal region of a single graph. The plot shows that the Bd implementation is considerably faster than the Precise implementation. For each graph class, we achieve on average a speed-up of at least 20. The minimum speed-up on the North graphs is 8. For each graph class, the speed-up is at least 18 for at least 75 out of 100 instances.

5.4 Random Sampling

The worst-case running time of computing the crossing-minimal region of a vertex v is super-quadratic in the size of the graph; see Figure 5.1. Figure 5.4 shows the number of intersecting segment in the arrangement $A(\Gamma, v)$ compared to the vertex-degree of

[1] http://graphdrawing.org/data.html

v, for vertices of three selected graphs with at most 2 133 edges, compare Table 5.1. For these graphs the arrangement already contains up to 440 685 519 intersecting segments. Indeed, we were not able to compute the number of intersections for all vertices of the graph c.metabolic, since the algorithm ran out of memory first. Due to the high number of intersections in graphs with a high number of edges or a large maximum vertex-degree, it is for these graphs infeasible to compute a crossing-minimal position of a vertex. This motivates the following question: Is a small subgraph of G sufficient to considerably reduce the number of crossings in a given drawing?

To address this question, we follow the *vertex-movement approach*. Let Γ_0 be a drawing of G and let $v_1, v_2, \dots, v_n$ be an ordered set of the vertices V of G. For each vertex v_i we obtain a new drawing Γ_i from the drawing Γ_{i-1} by moving v_i to a new position $p_i^\star$. To compute the new position we consider a *primal* sampling approach, i.e., a sampling of points in the solution space $\mathbb{R}^2$, and a *dual* sampling approach, i.e., a sampling of edges that induce constraints to the solution space.

More formally, we consider the following approach to compute a new position of a single vertex v_i. Let $S_i \subset E$ be a uniform random subset of the edges of G and let $V(S_i) \subset V$ be the vertices that are incident to an edge in S_i. The graph $G|_{S_i} = (V(S_i) \cup N(v_i) \cup \{v_i\}, S_i \cup E(v_i))$ induces a drawing $\Gamma|_{S_i}$ in Γ_{i-1}. Let R_i be the crossing-minimal region of v_i with respect to the drawing $\Gamma|_{S_i}$. Recall that for $S_i = E$ the region R_i has the property that $\mathrm{cr}(\Gamma|_{S_i}[v_i \mapsto p], v_i) = \mathrm{cr}(\Gamma|_{S_i}[v_i \mapsto q], v_i)$ for any two points $p, q \in R_i$, compare Section 5.2. If S_i is a strict subset of E, then R_i does not necessarily have this property anymore. For this reason, let $P_i \subset R_i$ be a set of uniform random points and let $p_i^\star \in P_i \cup \{p_i'\}$ be the point that minimizes $\mathrm{cr}(\Gamma[v \mapsto p_i^\star], v_i)$, where p_i' is the position of v_i in Γ_{i-1}.

This remainder of this section is organized as follows. First, we analyze the dual sampling from a theoretical perspective (Section 5.4.1), followed by an experimental evaluation that compares the primal to the dual sampling (Section 5.4.2). Finally, based on the insights from this evaluation, we introduce in Section 5.4.3 a *weighted* sampling approach that is less restrictive than the dual sampling.

5.4.1 Approximating the Co-Crossing Number of a Vertex

In this section we study the dual sampling approach, i.e., the sampling of edges, with tools introduced in the context of the theory of VC-dimension. A thorough introduction into the theory of VC-dimension can be found in Matoušek's *Lectures on Discrete Geometry* [Mat02]. We will prove that computing the optimal position of a vertex with respect to a small random subset of the edges is sufficient to approximate the (so-called) co-crossing number of a vertex v. This statement is only true for for drawings where the edges incident to v do not have to many crossings. We will formalize this as follows.

For a fixed vertex v, a drawing Γ is ε-*well behaved* if for each point $p \in \mathbb{R}^2$ and each vertex $u \in N(v)$, the edge uv crosses at most $(1-\varepsilon)|E|$ edges in the drawing $\Gamma[v \mapsto p]$. The *co-crossing number* co-cr(Γ, v) of a vertex v is the number of edge pairs $e \in E$ and $uv \in E$ that do not cross. We show that given an ε-well-behaved drawing Γ of a graph $G = (V, E)$ and a degree-k vertex v, a random sample $S \subset E$ of size $\Theta(k \log k)$ enables us to compute a position $q^\star$ whose co-crossing number is a $(1-\delta)$-approximation of the co-crossing number of a vertex v. Note that we are not able to guarantee that a large co-crossing number of a vertex v implies a small crossing number of v. On the other hand, the co-crossing number is of interest for a variety of (sparse) graph. For example, drawings that contain many triangles are ε-well-behaved, since every line intersects at most two segments of a triangle.

A *set system* is a tuple $(X, \mathcal{F})$ with a base set X and $\mathcal{F} \subseteq 2^X$. In the following, we assume X to be finite. For some parameters $\varepsilon, \delta \in (0, 1]$, a set $S \subseteq X$ is a *relative* (ε, δ)-*approximation* for the set system $(X, \mathcal{F})$ if for each $R \in \mathcal{F}$ the following inequality holds.

$$\left| \frac{|S \cap R|}{|S|} - \frac{|R|}{|X|} \right| \leq \delta \max\{\frac{|R|}{|X|}, \varepsilon\} \tag{5.1}$$

The following proposition states that, if each set R is sufficiently large, then a (ε, δ)-approximation S approximates each R. With the help of the concept of VC-dimension, we will show for our setting that there is a set S, whose size does not depend on the size of G, that is an (ε, δ)-approximation and such a set S can be computed with high probability.

Proposition 5.2. *For $\varepsilon, \delta \in (0, 1]$, let S be an (ε, δ)-approximation of the set system $(X, \mathcal{F})$. If every $R \in \mathcal{F}$ has size at least $\varepsilon|X|$ then Equation 5.1 can be rewritten as follows:*

$$(1-\delta)\,|R| \leq |X| \frac{|S \cap R|}{|S|} \leq (1+\delta)\,|R|\,.$$

Proof. In order to proof the claim, we make a case distinction based on the size of R. We first assume that $|S \cap X|/|S| < |R|/|X|$. Thus, we immediately get that $|X||S \cap R|/|S| \leq |R| \leq (1+\delta)|R|$ Moreover, the following holds $||S \cap R|/|S| - |R|/|X|| = |R|/|X| - |S \cap R|/|S|$. Starting from the fact S is (ε, δ)-approximation, we can do the following transformations.

$$\begin{aligned} |X| \left(\frac{|R|}{|X|} - \frac{|S \cap R|}{|S|} \right) &\leq \delta |X| \max \left\{ \frac{|R|}{|X|}, \varepsilon \right\} \\ \Leftrightarrow |R| - |X| \frac{|S \cap R|}{|S|} &\leq \delta \max\left\{|R|\,, \varepsilon\,|X|\right\} \\ \Leftrightarrow |X| \frac{|S \cap R|}{|S|} &\geq |R| - \delta\,|R| = (1-\delta)\,|R| \end{aligned}$$

In order to complete the proof, assume that $|S \cap X|/|S| \geq |R|/|X|$.

$$
\begin{aligned}
|X| \left(\frac{|S \cap R|}{|S|} - \frac{|R|}{|X|} \right) &\leq \delta |X| \max \left\{ \frac{|R|}{|X|}, \varepsilon \right\} \\
\Leftrightarrow |R| - |X| \frac{|S \cap R|}{|S|} &\leq \delta \max \{ |R|, \varepsilon |X| \} \\
\Leftrightarrow |X| \frac{|S \cap R|}{|S|} &\leq |R| + \delta |R| = (1 + \delta) |R|
\end{aligned}
$$

□

Let $\mathcal{F}|_A = \{R \cap A \mid R \in \mathcal{F}\}$ be the restriction of $\mathcal{F}$ to a set $A \subseteq X$. A set $A \subseteq X$ is *shattered by* $\mathcal{F}$ if every subset of A can be obtained by an intersection of A with a set $R \in \mathcal{F}$, i.e., $\mathcal{F}|_A = 2^A$. The *VC-dimension* of a set system $(X, \mathcal{F})$ is the size of the largest subset $A \subseteq X$ such that A is shattered by $\mathcal{F}$ [VC71].

Theorem 5.3 (Har-Peled and Sharir [HS11], Li et al. [LLS01]). *Let $(X, \mathcal{F})$ be a finite set system with VC-dimension d, and let $\delta, \varepsilon, \gamma \in (0, 1]$. A uniform random sample $S \subseteq X$ of size*

$$
\Theta \left(\frac{d \cdot \log \varepsilon^{-1} + \log \gamma^{-1}}{\varepsilon \delta^2} \right)
$$

is a relative (ε, δ)-approximation for $(X, \mathcal{F})$ with probability $(1 - \gamma)$.

For a vertex $u \in N(v)$, let $\overline{\mathbb{E}_{uv}(\Gamma)} = \{e \in E \mid \mathrm{cr}(\Gamma, e, uv) = 0\}$ denote the set of edges that are not crossed by the edge uv in Γ. Then we have $\text{co-cr}(\Gamma, v) = \sum_{u \in N(v)} \left| \overline{\mathbb{E}_{uv}(\Gamma)} \right|$. Moreover, let $\overline{\mathbb{E}_{uv}(p)} = \overline{\mathbb{E}_{uv}(\Gamma[v \mapsto p])}$. Then the set $\overline{\mathcal{F}_{uv}} = \bigcup_{p \in \mathbb{R}^2} \left\{ \overline{\mathbb{E}_{uv}(p)} \right\}$ contains for each drawing $\Gamma[v \mapsto p]$ the set of edges that are not crossed by the edges uv, i.e, $\overline{\mathbb{E}_{uv}(p)}$. In particular $(E, \overline{\mathcal{F}_{uv}})$ is a set system and we will prove that it has bounded VC-dimension. This allows us to approximate the number of edges that are not crossed by the edge uv. We facilitate this to approximate the co-crossing number of a vertex for ε-well behaved drawings.

Lemma 5.4. *The VC-dimension of the set system $(E, \overline{\mathcal{F}_{uv}})$ is at most 8.*

Proof. Recall that that vertex u has a fixed position. Let $\mathcal{BD}(u, e)$ be the boundary of the visibility region of u and the edge $e \in E$. Let A denote the arrangement of all boundaries $\mathcal{BD}(u, e)$, $e \in E$. Let F be the set of faces in A. Note that by Lemma 4.1 in Chapter 4 for every two points $p, q \in f$ the sets E_p and E_q of edges that have a non-empty intersection with the edge uv when v is moved to p and q, respectively, coincide. Hence, the set $E_f \subseteq E$ of edges that cross the edge uv, in the drawing obtained from Γ where v is moved to an arbitrary position in f, is well defined. Thus, the number of faces $|F|$ is an upper bound for $\left| \overline{\mathcal{F}_{uv}}|_A \right|$ for every $A \subset E$. Note that there may be subsets of E that are represented by more than one face. Moreover, observe that the visibility region $\mathcal{VR}(u, e)$ is the intersection of three half-planes. Let l_e^1, l_e^2, l_e^3 be the supporting lines of these half-planes and let A' be the arrangement of lines $l_e^i, e \in E$. Hence, the number of faces in the arrangement A' of $3m$ lines is an upper bound for $|F|$, with $m = |E|$. The number of faces $|F'|$ of A' is bounded by $f(m) := 3m(3m-1)/2 + 1$ [Moo91]. Thus, it is not possible to shatter a set $A \subset E$ if the number of faces $|F'|$ is smaller than the number of subsets of A. The largest number for which the equality $2^m \leq f(m)$ holds is between 8 and 9. Since 2^m grows faster than $f(m)$, the largest set that can possibly be shattered has size at most 8. □

Due to Proposition 5.2 and Theorem 5.3 a relative (ε, δ)-approximation S_u of $(E, \overline{\mathcal{F}_{uv}})$ allows us to approximate the number of edges that are not crossed by the edge uv. In the following we show that we can approximate the co-crossing number of a vertex v in any drawing $\Gamma[v \mapsto p]$ if we are given a relative (ε, δ)-approximation S_u for each vertex u that is adjacent to v. The number $|\overline{\mathbb{E}_{uv}(p)} \cap S_u|/|S_u|$ corresponds to the relative number of edges in S_u that are not crossed by the edge uv. Hence, the function $\lambda(p) = |E| \sum_{u \in U} |\overline{\mathbb{E}_{uv}(p)} \cap S_u|/|S_u|$ can be seen as an estimation of $\text{co-cr}(p) = \text{co-cr}(\Gamma[v \mapsto p], v)$.

Lemma 5.5. *Let $\varepsilon, \delta \in (0, 1]$ be two parameters and let Γ be an ε-well behaved drawing of G. For every $u \in N(v)$, let S_u be a relative (ε, δ)-approximation of the set system $(E, \overline{\mathcal{F}_{uv}})$. Then $(1 - \delta)\,\text{co-cr}(p) \leq \lambda(p) \leq (1 + \delta)\,\text{co-cr}(p)$ holds for all $p \in \mathbb{R}^2$.*

Proof. Recall that co-cr(p) is equal to $\sum_{u \in N(v)} |\overline{\mathbb{E}_{uv}(p)}|$. Since the drawing Γ is ε-well behaved, for every $u \in N(v)$ and every $p \in \mathbb{R}^2$ we have that at least an ε-fraction of edges is not crossed by the edge uv, i.e., $|\overline{\mathbb{E}_{uv}(p)}| \geq \varepsilon|E|$. Since S_u is a relative (ε, δ)-approximation and due to Proposition 5.2 we have that $(1-\delta)|\overline{\mathbb{E}_{uv}(p)}| \leq |E||\overline{\mathbb{E}_{uv}(p)} \cap S_u|/|S_u| \leq (1 + \delta)|\overline{\mathbb{E}_{uv}(p)}|$. Plugging this inequality into the sum of $\lambda(p)$ proves the lemma. □

Assume that $\varepsilon, \delta, \gamma \in (0, 1)$ are constants. Lemma 5.5 shows that k independent samples S_u of constant size approximate the co-crossing number of v. By slightly increasing the number of samples, we can use a single set S for all neighbors u. This reduces the running time from $O(k^3 \log k)$ to $O(k^2 \log^3 k)$.

Lemma 5.6. *Let v be a degree-k vertex and let $\varepsilon, \delta, \gamma \in (0, 1]$ with $\gamma \leq 1/k$. A uniformly random sample $S \subseteq E$ of size $\Theta\left((\log \varepsilon^{-1} + \log \gamma^{-1})/(\varepsilon\delta^2)\right)$ is a relative (ε, δ)-approximation the set system $(E, \overline{\mathcal{F}_{uv}})$ with probability $1 - k\gamma$, for each $uv \in E$.*

Proof. For each vertex $u \in N(v)$, we denote with A_u the event that S is a relative (ε, δ)-approximation of the set system $(E, \overline{\mathcal{F}_{uv}})$. According to Lemma 5.4 and Theorem 5.3 the probability $\mathbb{P}(A_u)$ that a uniformly random sample is a relative (ε, δ)-approximation of $(E, \overline{\mathcal{F}_{uv}})$ is $1-\gamma$. The following estimate can be proven by induction using the equalities $\mathbb{P}(A \wedge B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \vee B)$ and $\mathbb{P}(A \vee B) \leq 1$.

$$\mathbb{P}\left(\bigwedge_{u \in N(v)} A_u\right) \geq \sum_{u \in N(v)} \mathbb{P}(A_u) - k + 1$$

Plugging in the probability for $\mathbb{P}(A_u)$ proves that S is a relative (ε, δ)-approximation with probability $1 - k\gamma$ for a $\gamma \leq 1/k$. □

With Lemma 5.5 and Lemma 5.6 at hand, we have all the necessary tools to prove the main theorem.

Theorem 5.7. *Let $\varepsilon, \delta, \gamma \in (0,1]$ be three constants and let $G = (V, E)$ be a graph with a ε-well behaved drawing Γ and let $v \in V$ be a degree-k vertex. Let $p^\star$ be the position that maximizes* co-cr$(\Gamma[v \mapsto p^\star], v)$. *A $(1-\delta)$-approximation of* co-cr$(\Gamma[v \mapsto p^\star])$ *can be computed in $O\left(k^2 \log^3 k\right)$ time with probability $1-\gamma$.*

Proof. Let $\gamma' = \gamma \cdot k^{-1}$ and $\delta' = \delta/2$. Let $S \subseteq E$ be a uniformly random sample of size $\Theta\left((\log \varepsilon^{-1} + \log \gamma'^{-1})/(\varepsilon \delta'^2)\right)$. According to Lemma 5.6, for each $uv \in E$, the sample S is a (ε, δ')-approximation of the $(E, \overline{\mathcal{F}_{uv}})$ with probability $1 - k\gamma' = 1 - \gamma$.

According to Lemma 5.5 the expected number of crossing-free edges $\lambda(p)$ is a $(1-\delta)$-approximation of co-cr(p), i.e., $(1+\delta')\,\text{co-cr}(q) \geq \lambda(q) \geq (1-\delta')\,\text{co-cr}(q)$. Let $p^\star$ be the position that maximizes co-cr(p) and let $q^\star$ be the position that maximizes $\lambda(q)$. Hence, we have $\lambda(q^\star) \geq \lambda(p^\star)$. Observe that over $\delta' > 0$ the inequality $(1-\delta')/(1+\delta') \geq 1-2\delta'$ holds. We use this to prove that co-cr$(q^\star) \geq (1-2\delta')\,$co-cr$(p^\star)$.

$$\text{co-cr}(q^\star) \geq \frac{1}{(1+\delta')}\lambda(q^\star) \geq \frac{1}{(1+\delta')}\lambda(p^\star) \geq \frac{1-\delta'}{1+\delta'}\,\text{co-cr}(p^\star) \geq (1-2\delta')\,\text{co-cr}(p^\star)$$

Plugging in the value $\delta/2$ for δ' yields that co-cr$(q^\star)$ is a δ-approximation of co-cr$(p^\star)$. Since the three parameters $\varepsilon, \delta, \gamma$ are constants, the size of the sample S is in $\Theta(\log k)$. Recall that the running time to compute the crossing-minimal position of v in a drawing Γ is $O((kn+m)^2 \log(kn+m))$ (Theorem 5.1). Thus the position $q^\star$ can be computed in $O(k \log k + \log k)^2 \log(k \log k + \log k))$ time, since $m = |S| \in \Theta(\log k)$ and $n \leq 2m$. The following estimation concludes the proof.

$$O\left(k^2 \log^2 k \log(k \log k)\right) = O\left(k^2 \log^2 k \log(k^2)\right) = O(k^2 \log^3 k)$$

□

Note that the previous techniques can be used to design a δ-approximation algorithm for the crossing number of a vertex. But this requires drawings of graphs where at least $\varepsilon|E|$ edges, i.e., $\Omega(|E|)$, are crossed. This restriction is not too surprising, since sampling the set of edges can result in an arbitrarily bad approximation for a vertex whose crossing-minimal position induces no crossings.

5.4.2 Experimental Evaluation

In this section we complement the theoretical analyses of the random sampling of edges with an experimental evaluation. We first introduce our benchmark instances, followed by a description of a preprocessing step to reduce trivial cases and a set of configurations that we evaluate.

Benchmark Instances. We evaluate our algorithm on graphs from three different sources.

DIMACS The graphs from this classes are selected from the 10th Dimacs Implementation Challenge - Graph Partitioning and Graph Clustering [Bad+18].

Sparse MC Inspired by the selection of benchmark graphs in [MNS18], we selected a few arbitrary graphs from the Suite Sparse Matrix Collection (formerly known as the Florida Sparse Matrix Collection) [DH11].

k-**regular** For each $k = 3, 6, 9$ we computed 25 random k-regular graphs on 1000 vertices following the model of Steger and Wormald [SW99].

Preprocessing. Some of the benchmark graphs contain multiple connected components. Moreover, we observed that the Stress layout introduces crossings with edges that are incident to a degree-1 vertex. In both cases, these crossings can be removed. Therefore, we reduce the benchmark instances so that they do not contain these trivial cases as follows. First, we evaluate only the connected component G_C of each graph G that has the highest number of vertices. Further, we iteratively remove all vertices of degree 1 from G_C.

The vertex-movement approach takes an initial drawing of a graph as input. Note that the experimental results in Chapter 4 showed that drawings obtained with Stress have the smallest number of crossings compared to other energy-based methods implemented in Ogdf. In order to avoid side effects, we first computed a random drawing for each graph G_C where each coordinate is chosen uniformly at random on a grid of size $m \times m$. Afterwards we applied the Stress method implemented in Ogdf [Chi+13] (snapshot 2017-07-23) to this drawing.

Configurations. The previously described approach moves the vertices in a certain order. We use the order proposed in Chapter 4, i.e, in descending order with respect to the function $\mathrm{cr}(\Gamma_0, v_i)^2, v_i \in V$, where Γ_0 is the initial drawing. The computation of the new position $p_i^\star$ of a vertex v_i depends on three parameters $(|S_i|, |P_i|, K)$. The parameter K is a threshold on the degree k_i of v_i, since we observed in our preliminary experiments, that in case that k_i is large, $128\,GB$ of memory are not sufficient to compute the crossing-minimal region. Note that in case that $|S_i|$

Table 5.1: Statistics for the Dimacs and Sparse MC graphs. n, m, and $\overline{\Delta}$ correspond the number of vertices, edges and the mean vertex-degree, respectively.

	n	**m**	$\overline{\Delta}$	**crossings**			**time [min]**	
				Stress	$\mathcal{S}_{512}$	$\mathcal{S}_0$	$\mathcal{S}_{512}$	$\mathcal{S}_0$
Dimacs								
adjnoun	102	415	8.14	6 576	3 775	4 468	0.11	0.09
football	115	613	10.66	6 865	3 568	4 030	0.14	0.17
netscience	352	887	5.04	1 724	583	814	0.53	0.31
c.metabolic	445	2 017	9.07	113 117	55 714	63 028	11.29	2.29
c.neural	282	2 133	15.13	128 068	86 641	90 920	5.23	2.07
jazz	193	2 737	28.36	223 990	143 647	153 040	5.22	3.31
power	3 353	5 006	2.99	7 622	6 854	6 293	4.56	10.74
email	978	5 296	10.83	504 144	342 020	357 272	37.12	12.48
hep-th	4 786	12 766	5.33	836 809	546 780	638 069	72.86	78.24
Sparse MC								
1138_bus	671	991	2.95	657	402	467	0.41	0.33
ch7-6-b1	630	1 243	3.95	64 055	24 928	26 055	6.54	0.79
mk9-b2	1 260	3 774	5.99	412 397	248 884	252 198	20.33	7.14
bcsstk08	1 055	5 927	11.24	455 069	342 996	344 644	67.30	18.70
mahindas	1 258	7 513	11.94	1 463 437	933 247	1 042 787	68.17	24.09
eris1176	892	8 405	18.85	1 682 458	1 030 881	1 087 605	77.09	27.33
commanche	7 920	11 880	3.00	6 332	6 239	6 146	6.52	56.75

is constant the running time to compute R_i is $O((k_i \cdot n')^2 \log n') = O(k_i^2)$, where $n' = |V(S)| \in O(|S|)$. We handle vertices of degree larger than K, as follows. Let $N_1 \cup \cdots \cup N_l$ be a partition of the neighborhood $N(v)$ of v with $l = |N(v)|/K$. Further, let $u_1, u_2, \ldots, u_k$ be a random order of $N(v)$, then N_j contains the vertices u_a with $j \leq a \leq j + K$. For each j, we compute a random sample S_i^j and a crossing-minimal position $q_j^\star$ of vertex v with neighborhood N_j with respect to S_i^j. The new position $p_i^\star$ of v_i is the position that minimizes $\mathrm{cr}(\Gamma[v_i \mapsto q_j^\star], v_i)$.

We select the same parameters for each vertex and thus denote the triple by $(|S|, |P|, K)$. We expect that with an increasing number $|S|$ the number of crossings decreases. The sample size $|S| = 512$, was the largest number of samples such that we are able to compute a final drawing of our benchmark instances in reasonable time. As a baseline we sample 1000 points in the plane. Thus, we evaluate the following two configuration, $\mathcal{S}_{512} = (512, 1, 100)$ and $\mathcal{S}_0 = (0, 1000, \infty)$. Finally, we restrict the movement of a single vertex to be within an axis-aligned square that is twice the size of the smallest axis-aligned squares that entirely contains Γ_0.

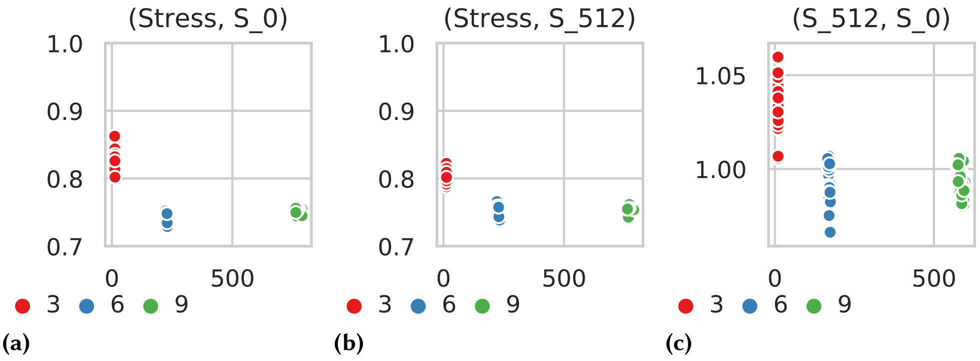


Figure 5.5: Number of crossings of the k-regular graphs.

Evaluation. Table 5.1 lists statistics for the Dimacs and the Sparse MC graphs. In particular the number of crossings of the initial drawing (Stress) and the drawing obtained by the $\mathcal{S}_{512}$ and $\mathcal{S}_0$ configurations. Furthermore, we report the running times for the two configurations. Since we use an external library (Ogdf) to compute the initial drawing, the reported times do not include the time to compute the initial drawing. Note that Stress required at most 0.9 min to complete on the Dimacs graph and 2.3 min on the Sparse MC graphs. Since the size of the arrangement $A(\Gamma, v)$ depends on the degree of v, the overall running time varies with the number of vertices and the average degree. Compare, e.g., c.metabolic to c.neural, or mk9-b2 to bcsstk08. Moreover, the commanche graph shows that the running time of $\mathcal{S}_0$ is not necessarily smaller than the running time of S_{512}. For each point $p \in P$ the number of crossings of edges incident to v in $\Gamma[v \mapsto p]$ have to be counted. Since the commanche graph contains over 11 000 edges, the $\mathcal{S}_{512}$ configuration with $|P| = 1$ is faster than the $\mathcal{S}_0$ configuration, which has to count the number of crossings for 1 000 points.

Now consider the number of crossings in the initial drawing (Stress) and in the drawing obtained by the $\mathcal{S}_{512}$ configuration. Since we move a vertex only if it decreases its number of crossings, it is expected that the number of crossings decreases on all graphs. For most graphs, the $\mathcal{S}_{512}$ configuration decreases the number of crossings by over 30%. In case of the ch7-6-b1 and the netscience graph the number of crossings are even decreased by over 60%. Exceptions are the bcsstk08, power and commanche graphs with 24%, 10% and 1.4% respectively. Comparing the number crossings obtained by $\mathcal{S}_{512}$ to the configuration $\mathcal{S}_0$, $\mathcal{S}_0$ results in fewer crossings only on two graphs (power, commanche).

Observe that the power, 11138_bus, ch7-6-b1 and commanche graphs all have an average vertex-degree of roughly 3.0. The comparison of the number of crossing obtained by $\mathcal{S}_{512}$ and $\mathcal{S}_0$ is not conclusive, since $\mathcal{S}_0$ yields fewer crossings on the power and commanche graphs and $\mathcal{S}_{512}$ on the remaining two. In order to be able to further

Table 5.2: Mean Number of crossings and standard deviation of number of crossings in drawings of the k-regular graphs computed by $\mathcal{S}_0$, $\mathcal{S}_{512}$ and Stress.

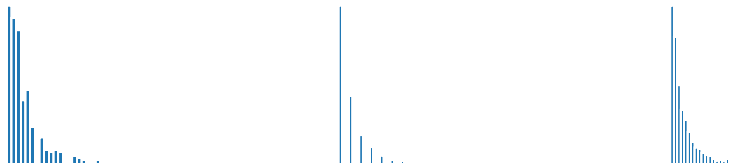
k	crossings $\mathcal{S}_0$		crossings $\mathcal{S}_{512}$		crossings Stress	
	mean	std	mean	std	mean	std
3	10 402.64	258.90	10 043.76	285.83	12 487.96	384.04
6	169 365.52	2260.86	170 558.48	2 379.56	227 303.68	3 450.72
9	580 661.80	6333.13	584 505.16	7 393.01	774 791.92	8 461.29

study the effect of the (average) vertex degree we evaluate the number of crossings of k-regular graphs. We use the plots in Figure 5.5 for the evaluation and Table 5.2 lists the corresponding descriptive statistics. Each point (x_G, y_G) corresponds to a k-regular graph G. The color encodes the vertex-degree. Let Γ_A and Γ_B be two drawings of G obtained by an algorithm A and B, respectively. The x-value x_G corresponds to the number of crossings in Γ_A in thousands, i.e., $\mathrm{cr}(\Gamma_A)/1000$. The y-value y_G is the quotient $\mathrm{cr}(\Gamma_B)/\mathrm{cr}(\Gamma_A)$. The titles of the plots are in the form (A, B) and encode the compared algorithms. For example in Figure 5.5a algorithm A is Stress and B is $\mathcal{S}_0$. For example, the Stress drawings of the 3-regular graphs have on average 12 487 crossings. Drawings obtained by $\mathcal{S}_0$ have on average 17% less crossings, i.e., 10 402; compare Table 5.2. On the other hand, $\mathcal{S}_{512}$ decreases the number of crossings on average by 20%. For $k = 6, 9$, $\mathcal{S}_0$ and $\mathcal{S}_{512}$ both reduce the number of crossings by 25%. In particular, Figure 5.5c shows that for $k = 6, 9$ it is unclear, whether $\mathcal{S}_{512}$ or $\mathcal{S}_0$ computes drawings with fewer crossings.

5.4.3 Weighted Sampling

For some graphs, the previous section gives first indications that sampling a set of edges yields a small number of crossings compared to a pure sampling of points in the plane. In particular Figure 5.5c indicates that the edge-sampling approach does not always have a clear advantage over sampling points in the plane. One reason for this might be that sampling within the set of points P_i in the region R_i is too restrictive. Observe that the region R_i is only crossing-minimal with respect to the sample S and does not necessarily contain the crossing-minimal position $p_i^\star$ of the vertex v_i with respect to all edges E. On the other hand, sampling the set of points P_i in $\mathbb{R}^2$ does not use the structure of the graph at all. This motivates the following *weighted* approach of sampling points in $\mathbb{R}^2$.

For a set $S \subset E$, let cr_j be the number of crossings of the vertex v_i with respect to $\Gamma|_S$, when v_i is moved to a cell c_j of the arrangement $A(\Gamma|_S, v_i)$. Let M be the maximum of all cr_j. We select a cell c_j with the probability $2^{M-\mathrm{cr}_j}/\sum_k 2^{M-\mathrm{cr}_k}$. Within a given cell, we draw a point uniformly at random. Note that in case that there are exactly



(a) netscience **(b)** power **(c)** hep-th

Figure 5.6: Degree distribution of a selection of graphs on which the $\mathcal{W}_{512}$ computes a small number of crossings.

n cells such that cell c_j induces j crossings, the probability that the cell c_0 is drawn converges to $1/2$ for $n \to \infty$.

Benchmark Instances, Preprocessing & Methodology. We use the same set of benchmark instances and the same preprocessing steps as described in Section 5.4. In order to obtain more reliable results, we perform 10 independent iterations for each configuration on the Dimacs and Sparse MC graphs. Since the k-regular graphs are uniform randomly computed, they are already representative for their class. Therefore, we perform only single runs on these graphs.

Configuration. We compare the following three configurations. $\mathcal{R}_0$ refers to the uniform random sampling of points in $\mathbb{R}^2$ with the parameters $(|S|, |P|, K) = (0, 1000, \infty)$, $\mathcal{R}_{512}$ to the restricted sampling in R_i with the parameters, $(512, 1000, 100)$, and $\mathcal{W}_{512}$ to the weighted sampling in $\mathbb{R}^2$ with the parameters $(512, 1000, 100)$. The configurations are selected such that $\mathcal{R}_0$ and $\mathcal{R}_{512}$ differ only in a single parameter, i.e., in the number of sampled edges. The only difference between $\mathcal{R}_{512}$ and $\mathcal{W}_{512}$ is the sampling strategy. Note that the parameters of $\mathcal{R}_0$ and $\mathcal{S}_0$ coincide, but not the parameters of $\mathcal{S}_{512}$ and $\mathcal{R}_{512}$.

Evaluation. Since we executed 10 independent runs of the algorithm on each graph, Table 5.3 lists the mean and standard deviation of the computed number of crossings for each graph. For each graph, we marked the cell with the lowest number of crossings in green and the largest number in blue. For each graph, we used the Mann-Witney-U test [She03] to check the null hypothesis that the crossing numbers belong to the same distribution. The test indicates that we can reject the null hypothesis at a significance level of $\alpha = 0.01$, for all graphs with the exception of football, ch7-6-b1 and bcsstk08. First, observe that the $\mathcal{R}_0$ configuration never computes a drawing with fewer crossings than $\mathcal{R}_{512}$. Including the football, ch7-6-b1 and the bcsstk08

Table 5.3: Mean and standard deviation (std) of the number of crossings categorized by configuration. For each graph the configuration with the lowest and highest number of crossings in marked.

	$\mathcal{R}_0$ mean	$\mathcal{R}_0$ std	$\mathcal{R}_{512}$ mean	$\mathcal{R}_{512}$ std	$\mathcal{W}_{512}$ mean	$\mathcal{W}_{512}$ std
Dimacs						
adjnoun	4 445.0	39.55	3 655.7	62.96	3 951.2	19.53
football	3 973.6	97.93	3 350.0	83.38	3 247.0	73.84
netscience	819.0	30.73	497.1	28.78	437.8	12.87
c.metabolic	62 170.4	760.47	56 032.3	1 227.23	62 987.9	1 907.64
c.neural	89 744.3	1 239.22	86 500.8	1 364.5	99 426.1	1 258.98
jazz	152 013.8	1 930.13	147 387.1	3 134.15	213 019.4	1 696.07
power	6 301.1	33.51	4 512.8	63.09	3 912.5	30.97
email	356 583.4	3 512.0	341 503.8	3 480.74	351 168.7	2 624.18
hep-th	640 515.2	3 443.22	515 109.1	3 983.23	392 189.7	1 551.53
Sparse MC						
1138_bus	474.6	13.25	342.9	12.91	247.6	9.8
ch7-6-b1	25 874.7	356.58	25 172.4	582.48	28 443.5	960.3
mk9-b2	251 360.9	1 514.05	245 447.4	2 914.18	228 794.5	2 069.96
bcsstk08	346 404.4	3 730.3	328 182.0	6 127.69	330 213.8	1 726.01
mahindas	1 036 745.7	11 494.88	936 889.0	11 207.34	1 105 850.9	10 185.51
eris1176	1 103 184.6	21 475.11	1 037 509.5	29 877.3	1 492 423.4	25 457.93
commanche	6 135.2	13.08	5 370.3	24.75	5 979.4	14.72

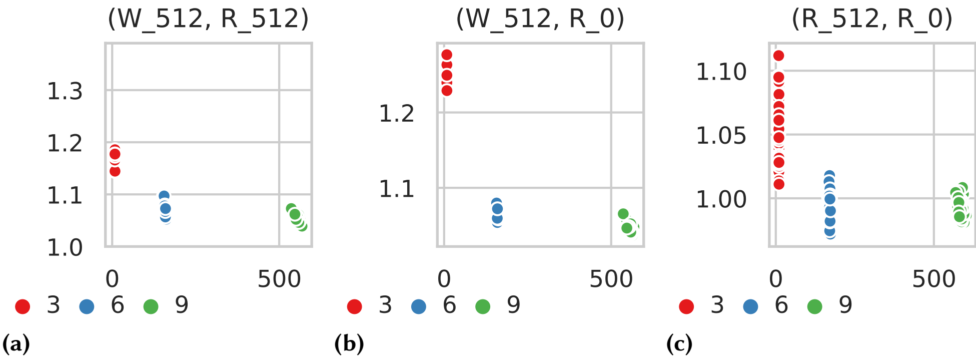


Figure 5.7: Comparison of the number of crossings of the k-regular graphs.

graphs, eleven of the drawings with the fewest crossings were obtained from the $\mathcal{R}_{512}$ configurations. Only seven correspond to the $\mathcal{W}_{512}$ configuration. Table 5.1 shows that these graphs have an average vertex-degree of at most 11. Moreover, the degree-distributions of these graphs follow the power-law. For an example, refer to Figure 5.6. On the other hand, a few of the graphs where $\mathcal{R}_{512}$ outperforms $\mathcal{W}_{512}$ also have a small average vertex-degree.

We use Figure 5.7 to compare the effect of the vertex-degree on the number of crossings. The plot follows the same convention as the plots in Figure 5.5. Observe that for each k, the $\mathcal{W}_{512}$ configuration computes drawings with fewer crossings than $\mathcal{R}_{512}$. The improvement decreases with an increasing k. The same observation can be made for the comparison of $\mathcal{W}_{512}$ to $\mathcal{R}_0$ but not for the comparison for $\mathcal{R}_{512}$ to $\mathcal{R}_0$, which indicates that sampling the set of points P_i within the region R_i is indeed too restrictive, at least on our k-regular graphs.

Overall our experimental evaluation shows that even with a naive uniform random sampling of a set of points in the plane the number of crossings in drawings of Stress can be reduced considerably. Using a random sample of a subset of the edges helps to compute drawings with even less crossings. The mean-vertex degree and the degree-distributions are good indicators for whether the restrictive or the weighted sampling of the point set P_i results in a drawing with the smallest number of crossings.

5.5 Conclusion

In our previous work we showed that the primitive operation of moving a single vertex to its crossing-minimal position significantly reduces the number of crossings compared to drawings obtained by Stress. In this chapter we introduced the concept of *bloated dual of line arrangements*, a combinatorial technique to compute a dual representation of line arrangements. In our applications of computing drawings with a small number of crossings, this technique resulted in a speed-up of factor of 20. This improvement was necessary to adapt the approach for graphs with a large number of vertices and edges. On the other hand, since the worst-case running time is super-quadratic, this improvement is not sufficient to cope with large graphs. In Section 5.4 we showed that random sampling is a promising technique to minimize crossings in geometric drawings. In Section 5.4.1 we proved that a random subset of edges of size $\Theta(k \log k)$ approximates the co-crossing number of a vertex v with a high high probability. Further, we evaluated three different strategies to sample a set of points in the plane in order to compute a new position for the vertex v_i. First, the evaluation confirms that the number of crossings compared to Stress can be reduced considerably. Furthermore, sampling a small subset of the edges is sufficient to reduce the number of crossings compared to a naive sampling of points the plane. Our evaluation suggests that weighted sampling is a promising approach to reduce the number of crossings in graphs with a low average vertex degree. Otherwise, the evaluation indicates that restricted sampling results in fewer crossings.

The running time of the vertex-movement approach in combination with the sampling of the edges mostly depends on the number of vertices. Since a single movement of a vertex is not optimal anymore, two vertices can be moved independently. Thus, future research should be concerned with the question whether a parallelization over the vertex set is able to further reduce the running time while preserving the small number of crossings. Moreover, we ask whether it is sufficient to move a small subset of the vertices to considerably reduce the number of crossings.

6 Stretching Topological Drawings

We study the problem of computing straight-line drawings of non-planar graphs with few crossings. We assume that a crossing-minimization algorithm is applied first, yielding a *planarization*, i.e., a planar graph with a dummy vertex for each crossing, that fixes the topology of the resulting drawing. We present and evaluate two different approaches for drawing a planarization in such a way that the edges of the input graph are as straight as possible. The first approach is based on the planarity-preserving force-directed algorithm ImPrEd [Sim+11], the second approach, which we call *Geometric Planarization Drawing*, iteratively moves vertices to their locally optimal positions in the given initial drawing.

Our evaluation shows that both approaches significantly improve the initial drawing and that our geometric approach outperforms the force-directed approach. To the best of our knowledge, this is the first approach that targets towards the computation of a straight-line drawing that respects an arbitrary planarization.

This chapter extends work initiated in my master thesis [Rad15] and is joint work with Thomas Bläsius and Ignaz Rutter [BRR17, BRR19].

6.1 Introduction

In his seminal paper "How to Draw a Graph" [Tut63], Tutte showed that every planar graph admits a planar straight-line drawing. His result has been strengthened in various ways, e.g., by improving the running time, the required area [Cha+12] or to restrict the position of some vertices to points on a line; compare Chapter 10 and [Da +18]. In practice, however, many graphs are non-planar and we are interested in finding straight-line drawings with few crossings. Unfortunately, crossing minimization for straight-line drawings is $\exists\mathbb{R}$-complete, i.e., as hard as the existential theory of the reals [Sch10]. We thus need to relax either the condition of minimizing the number of crossings or the requirement of straight edges. Approximating the rectilinear crossing number seems difficult, and for complete graphs K_n, it is only known for $n \leq 27$ [Ábr+08]. In Chapter 4 and Chapter 5, we require straight-line edges and heuristically minimize the number of crossings. In this chapter, we follow the second approach, i.e., we insist on a small (though not necessarily minimum) number of crossings and optimize the straightness of the edges in the drawing.

In contrast to the geometric setting, the crossing number for topological drawings has received considerable attention and there is a plethora of results on crossing minimization; see [Buc+13] for a survey. The output of these algorithms typically is a planarization G_p of the input graph G together with a planar embedding. To profit from the results in this area, we focus on the problem of drawing G_p such that for each edge of G the corresponding *planarization path* in the drawing of G_p is as straight as possible.

This type of problem is prototypical for several fundamental problems in graph drawing that ask for a geometric realization of a given combinatorial description of a drawing. The most prominent examples are the topology-shape-metrics framework for orthogonal graph drawing [Tam87] and the fundamental ($\exists\mathbb{R}$-complete) problem Stretchability, which asks whether a given arrangement of pseudolines can be realized by geometric lines [Mnë88]. There have been several other works that consider the problem of realizing a given combinatorial description of a drawing geometrically.

Thomassen [Tho88] gives a characterization for 1-planar graphs that admit a straight-line drawing. Moreover, he shows that there is no finite number of forbidden configurations that characterize the straight-line drawable 2-planar graphs. Di Giacomo et al. show that if the set of edges without crossings of a non-planar graph form a connected subgraph then there is a drawing of the same graph with at most three bends per edge that respects prescribed topological constraints [Gia+18]. Otherwise, the number of bends is in $\Omega(\sqrt{n}))$, where n is the number of vertices of G. Eades et al. study when a (maximal) planar graph with an additional edge has straight-line drawing [Ead+15]. In Chapter 8, we consider the problem of computing such a realizable embedding of a planar graph with an additional edge with a minimal number of crossings for restricted planar graph classes.

Chan et al. [Cha+15] prove that a linear number of bends per edges is sufficient to extend a given straight-line drawing of a planar graph. Given a fixed convex drawing of a face f of a planar graph, Mchedlidze et al. [MNR16] introduce a linear-time algorithm to test whether there is a straight-line drawing of a planar graph that extends the drawing of f. Grilli et al. [Gri+14] study the problem of realizing a given simultaneous planar embedding of two (or more) graphs with few bends per edge. For a survey on graph drawing beyond planarity see [DLM19].

The algorithm of Dwyer et al. [DMW09] minimizes the stress of a layout while preserving the topology of the drawing. Didimo et al. [DLR11] present an algorithm that is able to preserve the topology unless changing the topology improves the number of crossings. Bertault [Ber00] presents PrEd, a force-directed layout algorithm for planar graphs that preserves the combinatorial embedding of the input drawing; the approach was later improved by Simonetto et al. [Sim+11]. To the best of our knowledge the problem of producing a drawing of an arbitrary planarization such that the planarization paths are drawn as straight as possible has not been investigated prior to this work.

Contribution and Outline.. We study the problem of finding a drawing of a given planarization G_p of a graph G such that the planarization paths corresponding to the edges of G are drawn as straight as possible. We present two approaches, one is based on an adaption of ImPrEd that includes additional forces to facilitate straightening the planarization paths (Section 6.3). The second is a geometric framework that iteratively moves the vertices of a given drawing one by one to locally optimal positions such that (i) the planarization and its planar embedding are preserved and (ii) the angles on planarization paths influenced by that vertex are optimized (Section 6.4). This framework has several degrees of freedom, such as the vertex processing order and the exact placement strategy for vertices. We experimentally evaluate the modified ImPrEd algorithm (ImPrEd++) and several configurations of the Geometric Planarization Drawing approach in a quantitative study (Section 6.5). We show that all our methods significantly increase the straightness compared to the initial drawing and that the geometric algorithms typically outperform ImPrEd++ in terms of quality. Statistical tests are used to show that these results are significant with 95% confidence.

6.2 Preliminaries

Intuitively, a planarization of a graph G is the graph resulting from placing dummy vertices at the intersections of edges in a drawing of G. More formally, let $G = (V, E)$ be a graph and let $G_p = (V \dot\cup V_p, E' \dot\cup E_p)$ be a planar graph such that every edge in E_p is incident to at least one vertex in V_p. The vertices in V_p are called *dummy vertices*. Then G_p is a *planarization* of G if the following conditions hold. (i) Dummy vertices have degree 4, (ii) $E' \subseteq E$, (iii) for every edge $e = uw \in E \setminus E'$, G_p contains a *planarization path* from u to w whose edges are in E_p and whose internal vertices are in V_p, (iv) for any two distinct edges $e, e' \in E \setminus E'$ the paths p_e and $p_{e'}$ are edge-disjoint, and (v) the paths p_e, $e \in E \setminus E'$ cover all edges in E_p. We call the planarization G_p *k-planar* if the longest planarization path has k dummy vertices, i.e., there are at most k crossings per edge.

A *dissected pair* (u, v, w) is a pair $uv, vw \in E_p$ of edges that belong to the same planarization path; see Figure 6.1a. Note that formally (u, v, w) and (w, v, u) do not coincide but we for the purpose of this chapter we consider the two dissected pairs to be the same. The *straight-line-deviation angle* sd-$\alpha(u, v, w)$ of (u, v, w) is the angle sd-$\alpha(u, v, w) = \pi - \angle(u, v, w)$. We simply refer to a straight-line-deviation angle as *deviation angle*. A deviation angle is *active* with respect to v (also called *v-active*) if moving v can alter that angle. This notation allows us to formalize our problem of drawing the planarization paths of G_p as straight as possible as follows: Given an embedded planarization G_p of G and an angle α, is there a planar straight-line drawing of G_p with the given embedding such that all deviations angles are smaller than α, i.e.,

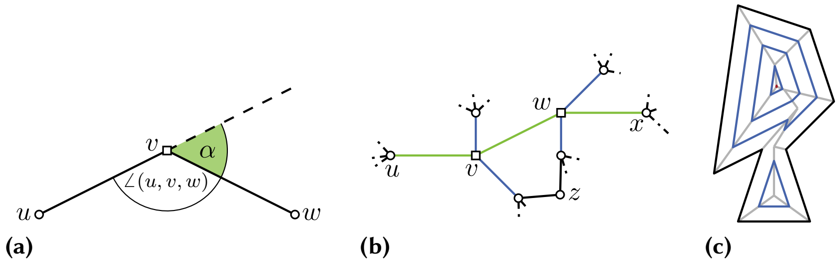


Figure 6.1: (a) The deviation angle sd-$\alpha(u, v, w) = \alpha$ of the dissected pair (u, v, w). (b) Vertex u and w are tail vertices of the dissected pair (u, v, w). Since w is a dummy vertex of the dissected pair (v, w, x), w is a hybrid vertex. z is an independent vertex. (c) A (grey) straight skeleton of a (black) polygon and a set of (blue) shrinked polygons. The geometric center is depicted in red.

sd-$\alpha(u, v, w) \le \alpha$ for every dissected pair (u, v, w) of G_p? The respective optimization problem asks for the minimum angle α.

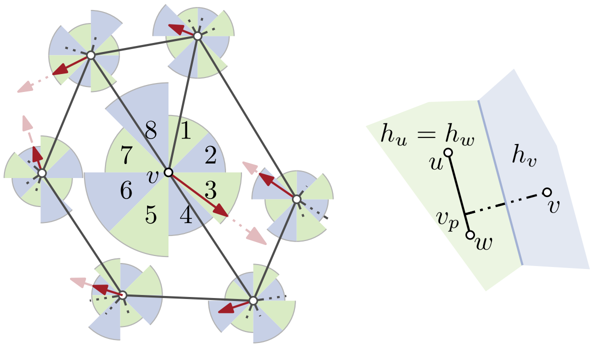
For a dissected pair (u, v, w), v is a dummy vertex and u and w are *tail* vertices; see Figure 6.1b. A dummy that is not a tail is called *pure dummy* and a tail that is not a dummy is called *pure tail.* Vertices that are both, tail and dummy, are called *hybrid.* A vertex that is neither a dummy nor a tail vertex is called *independent.*

Let P be a polygon and let v be a vertex of P. A point p in the interior of P is *visible* from v if the straight line connecting p with v does not intersect an edge of P. The *visibility region of v in P* is the set of all points in P that are visible from v. The *size* of a polygon P is the number of its vertices.

A *shrinked* polygon P' of a polygon P is the result of moving the vertices towards the interior of a polygon P with constant speed along the *straight skeleton* of P [HH11]; see Figure 6.1c. A *geometric center* of a polygon P is obtained by shrinking P to a single point. In case that the shrinking process yields disconnected polygons, we consider the center of the polygon with the largest area as the center of P.

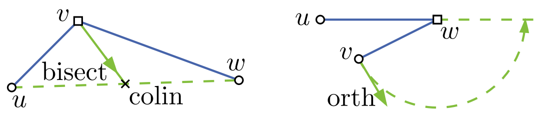
6.3 Force-Directed Planarization Drawing

We present a force-directed approach ImPrEd++ for straightening the planarization paths in a given drawing based on ImPrEd [Sim+11], a spring embedder that is able to preserve the planar embedding of a given drawing. ImPrEd preserves the combinatorial embedding of a planar straight-line drawing as follows. Let $Z_1, \ldots, Z_8$ be a partition of the unit disk around a vertex v into eight octants; refer to Figure 6.2a. The radius of each octant Z_i is scaled by a value R_i such that any movement of v by a direction lying inside Z_i preserves the combinatorial embedding. In order to allow a more flexible movement of each vertex, we substitute the radial zones with a convex polygon



(a) **(b)** **(c)**

Figure 6.2: (a) Radial zones (blue and green) used by ImPrEd. Forces (light red) are cropped at the boundary of the zones (dark red) (b,c) Construction of the half-planes L_x in case (b) that the projection of v lies on uv or (c) the projection does not lie on uv.



(a) **(b)**

Figure 6.3: Our new forces. If v is a dummy vertex (a), move it along the bisector of the adjacent segments. If v is a tail vertex (b), move it gradually along an arc.

P_v. The polygon corresponds to the construction given in the correctness proof of ImPrEd [Sim+11].

For each vertex v, let L_v be a set of half-planes constructed as follows; see Figure 6.2b and Figure 6.2c. For each edge uw of G and let v_p^{uw} be the projection onto the line through uw. If the projection v_p^{uw} does not lie on the segment uw, set v_p^{uw} to the closest point on uw. Let l_v be the line perpendicular to the segment vv_p^{uw} through the middle point of the segment vv_p^{uw}. For each vertex $x \in \{u, v, w\}$ we add the half-plane h_x of l_v that contains x to the set L_x. Finally, the polygon P_v is the intersection of all half-planes in the set L_v.

To reduce the deviation angles, we introduce the new forces F^d for dummy vertices and F^t for tail vertices. Hybrid vertices are affected by both forces. For independent vertices, we apply the same forces as ImPrEd.

Let v be a dummy vertex and let (u, v, w) be a dissected pair containing v. To encourage placing v collinearly between u and w, we apply a force in the direction of the unit length bisector bisect(u, v, w) of the vectors $u - v$ and $w - v$; see Figure 6.3a.

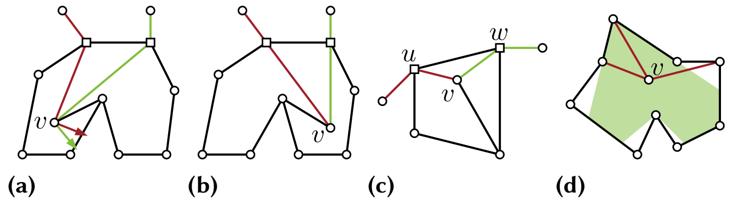


Figure 6.4: An initial drawing (a) that is difficult to repair using the force-directed algorithm although v could be moved to an optimal position without violating planarity (b). (c) The closer v lies to the edge uw, the better are the v-active angles. (d) The (green) planarity region of v.

Let colin(u, v, w) denote the point on the bisector that is collinear with u and w. We use the dummy force $F^d(v, (u, w)) = \lambda(\text{colin}(u, v, w) - v)$, where $0 < \lambda < 1$ is a damping factor. To form the dummy force $F^d(v)$ for v, we sum over the two dissected pairs where v is the dummy vertex.

For a tail vertex v and a dissected pair (u, w, v), we want to place v on the extension of the segment uw; see Figure 6.3b. To accomplish this, we try to perform a radial movement of v around w over several iterations of the spring embedder. Hence, we introduce a force in the normalized direction orth(u, w, v) of the tangent at v with the circle centered at w and passing through v. The direction of orth(u, w, v) is chosen such that it points away from the segment uw. The strength of the force is proportional to dist(v, w) with a damping factor of $0 < \kappa < 1$, i.e., $F^t(v, (u, w)) = \kappa\,\text{dist}(w, v)\,\text{orth}(u, v, w)$. To obtain the resulting force for a tail vertex v, we sum over all dissected pairs where v is a tail vertex.

6.4 Geometric Planarization Drawing

The spring embedder described in Section 6.3 restricts the movement of each vertex in a very conservative manner, i.e., the restrictions ensure a preservation of the given planar embedding. This may waste a lot of potential; see Figure 6.4a and Figure 6.4b. The approach presented in this section aims to tap the full potential by making each movement locally optimal. As the simultaneous movement of multiple vertices leads to non-trivial and non-local dependencies, we move only a single vertex in each step.

To make this precise, we need to answer two questions. First, to which points can a vertex v be moved such that the planar embedding is preserved? Second, which of these points is the best position for v? Concerning the first question, we call the set of points satisfying this property the *planarity region* of v and denote it by $\mathcal{PR}(v)$. We show in Section 6.4.1 how to compute $\mathcal{PR}(v)$ efficiently. Concerning the second question, we define the *cost* of a point $p \in \mathcal{PR}(v)$ to be the maximum of all v-active deviation angles when placing v at p. A point in $\mathcal{PR}(v)$ is a *locally optimal* position for v if $\mathcal{PR}(v)$ contains no other point with strictly smaller cost. In Section 6.4.2, we show how to compute an arbitrarily exact approximation of the locally optimal position.

The overall algorithm can be described as follows. We iterate over all vertices of the graph. In each step, the current vertex is moved to its locally optimal position. We repeat until we reach a drawing that is stable or a given number of iterations is exceeded.

One important degree of freedom in this algorithm is the order in which we iterate over the vertices. Another choice we have not fixed so far is the placement of independent vertices. As an independent vertex has no active angle, each point in its planarity region is equally good. We propose and evaluate different ways of filling these degrees of freedom in Section 6.5.

For a tail or dummy vertex v, it can happen that there exists no locally optimal position due to the fact that $\mathcal{PR}(v)$ is an open set. The cost may for example go down, the closer we place v to an edge connecting two other vertices; see Figure 6.4c. We therefore shrink $\mathcal{PR}(v)$ slightly and consider it to be a closed set. On one hand, this ensures that a locally optimal position always exists. On the other hand, it (partially) prevents that vertices are placed too close to edges, which is usually not desirable in a drawing. The offset by which we shrink $\mathcal{PR}(v)$ is discussed in Section 6.5, where we describe our exact evaluation setup.

6.4.1 Planarity Region

Let G_p be a planarization with a given drawing and let v be a vertex of G_p. Let f_v be the face of $G_p - v$ that contains the current position of v. Assume for now that f_v is bounded by a polygon $\mathrm{surr}(v)$, which we call the *surrounding* of v. Consider a point p in the interior of f_v and assume that we use p as the new position for v. Clearly, the resulting drawing is planar if and only if p is visible from each of v's neighbors; see Figure 6.4d.

Thus, the planarity region $\mathcal{PR}(v)$ is the intersection of all visibility regions in $\mathrm{surr}(v)$ with respect to the neighbors of v. It follows that the planarity region can be obtained by first computing the visibility polygons of v's neighbors in $\mathrm{surr}(v)$, and then intersecting these visibility polygons. Let n_v be the number of vertices of the surrounding polygon $\mathrm{surr}(v)$ and let d_v be the degree of v. Observe that if v is not a cut-vertex then $\mathrm{surr}(v)$ does not have holes and computing the d_v visibility polygons takes $O(d_v n_v)$ time [JS87]. To intersect these d_v visibility polygons (each having size $O(n_v)$), one can use a sweep-line algorithm [NP82] consuming $O((k + d_v n_v)\log n_v)$ time, where k is the number of intersections between segments of the visibility polygons. As there are at most $d_v n_v$ segments, $k \in O(d_v^2 n_v^2)$ holds, yielding the running time $O(d_v^2 n_v^2 \log n_v)$ for computing the planarity region. We first show that we can improve this running time in case that v is not a cut-vertex. Subsequently, we show how to modify $\mathrm{surr}(v)$ so that we are able to apply the following lemma.

Lemma 6.1. *If v is not a cut-vertex, then the planarity region $\mathcal{PR}(v)$ of v has size $O(n_v)$ and can be computed in $O(d_v n_v \log n_v)$ time.*

Proof. Let P_u be the visibility polygon of u in $\mathrm{surr}(v)$. A segment w on the boundary of P_u that is not part of a segment of $\mathrm{surr}(v)$ is called *window*. We say that the window w is *generated* by u; compare Figure 6.5. Instead of intersecting the visibility polygons of all neighbors, we compute the planar subdivision induced by the segments of $\mathrm{surr}(v)$ and all windows generated by neighbors of v. As there are only $O(d_v n_v)$ windows, this can be done (again using a simple sweep-line algorithm) in $O((k + d_v n_v) \log n_v)$ time, where k is the number of intersections between segments, i.e., the number of vertices of the resulting planar subdivision H. We show the following three claims.

Claim 4. *The planarity region of v is a face of the subdivision H.*

Claim 5. *Every window intersects with $O(d_v)$ segments.*

Claim 6. *It suffices to consider $O(n_v)$ windows.*

The first claim implies that we can compute the planarity region in linear time in the size of H as we only need to find the face of H containing the previous position of v (which is clearly contained in the planarity region). Each vertex of H is either a vertex of $\mathrm{surr}(v)$ or an intersection of a window with a segment (which is either also a window or a segment of $\mathrm{surr}(v)$). Thus, the second and third claim show that $k \in O(d_v n_v)$ holds. It is moreover not hard to see that no two different edges on the boundary of a face of H belong to the same segment of $\mathrm{surr}(v)$ or to the same window. Thus, each face (and in particular the planarity region) is bounded by only $O(n_v)$ edges, which concludes the proof.

To prove Claim 4, first note that $\mathrm{surr}(v)$ is the outer face of H, as every window lies completely inside $\mathrm{surr}(v)$. Let f be the face of H containing the previous position of v. We step by step remove subgraphs of H that eliminate only faces that cannot be part of the planarity region $\mathcal{PR}(v)$. In the end, only the face f remains, which shows $\mathcal{PR}(v) = f$. For this purpose consider an edge e incident to f. If e is not on the outer face of H, then e is part of a window w. We can extend e to a path π between vertices on the outer face such that the edges on π are all part of w. Then π separates H into two parts. Faces in the part not containing f clearly cannot be part of the planarity region due to the window w. Thus, we can remove this part, which has the effect that e now lies on the outer face. Once all edges incident to f lie on the outer face, the claim follows.

For Claim 5, observe that every window has two intersections with segments of $\mathrm{surr}(v)$. Thus, all remaining intersections are with other windows. Let w_1 be a window generated by the neighbor u_1 of v and let u_2 be another neighbor of v. We show that w_1

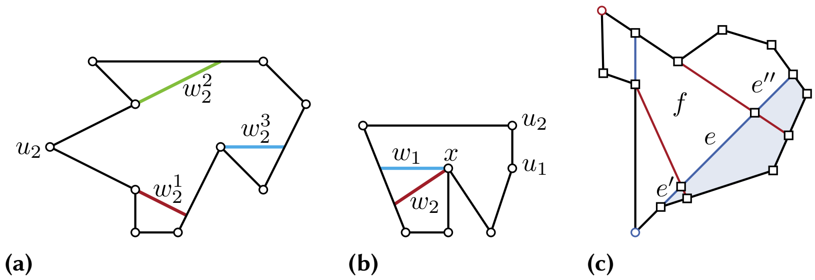


Figure 6.5: (a) Three windows generated by the neighbor u_2. (b) The window w_2 (generated by u_2) is dominated by w_1 (generated by u_1). (c) The edges e' and e'' extend e to a path π that correspond to a window of the neighbor u_1 (blue). Removing the blue region that does not contain f, reduces the size of H (squared vertices).

intersects at most two windows generated by u_2, which directly implies the claim. To this end, consider three windows w_2^1, w_2^2, and w_2^3 generated by u_2; see Figure 6.5a. Since the lines through w_i^2 intersect in u_2, the planar subdivision of surr(v) with these three windows has four inner faces; one face incident to all three windows (and to edges of surr(v)), and one face for each window w_2^i (for $i \in \{1, 2, 3\}$) that is only incident to w_2^i and edges of surr(v). A window w_1 intersecting all three windows w_2^1, w_2^2, and w_2^3 would need to cross the boundary of each of the latter three faces exactly once, which is clearly impossible. Thus, w_1 can intersect at most two windows generated by u_2.

To show Claim 6, note that at least one endpoint of every window is a concave corner in surr(v), i.e., a vertex of surr(v) with an interior angle that is grater than 180°. Consider one concave corner x and let w_1 and w_2 be two windows with endpoint x. The window w_1 separates surr(v) into two parts, one of which cannot be part of the planarity region. If w_2 lies in this part, then w_2 yields no real restriction compared to w_1; see Figure 6.5b. Thus, we say that w_2 is *dominated* by w_1. Clearly, removing all dominated windows does not alter the result of the algorithm. Moreover, it is not hard to see that there can be at most two non-dominated windows sharing an endpoint. Thus, Claim 6 follows, which concludes the proof. □

Theorem 6.2. *The planarity region can be computed in $O(d_v^2 n_v \log n_v)$ time.*

Proof. If v is not a cut-vertex, we can apply Lemma 6.1. Hence, consider the case that v is a cut-vertex. Then the surrounding polygon surr(v) has holes. In the following, we show how to locally modify G such that v is not a cut-vertex anymore and such that the planarity region of v in the new graph coincides with the planarity region of v in G. Let $P_0, P_1, \ldots, P_k$ be the polygons that describe the boundary of surr(v), i.e., P_0 is the outer polygon and $P_1, \ldots, P_k$ the holes in the interior of P_0; see Figure 6.6. Moreover, let u_i be a neighbor of v that lies on the boundary of P_i. Consider the ray R_i starting in

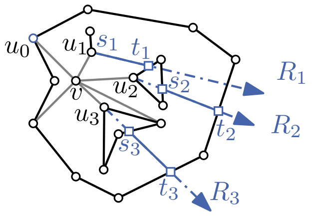


Figure 6.6: The blue segments are added as edges to G to ensure that v is not a cut-vertex.

u_i in the direction from u_0 towards u_i. Let $s_i t_i \in R_i$ be the segment of minimal length such that s_i lies on P_i and t_i on $P_j, j \neq i$. We subdivide the corresponding edges in G and add $s_i t_i$ as an edge to G.

Clearly the planarity region of v in the modified graph and the original graph coincide and v is not a cut-vertex anymore. To finish the proof of the theorem we have to prove the claimed running time. First, the polygonal chains $P_0, \ldots, P_k$ and the neighbors u_i can be computed in $O(\sum_{i=0}^{k} |P_i|) = O(n_v)$ time. Each segment $s_i t_i$ can be computed in $O(\sum_{i=0}^{k} |P_i|) = O(n_v)$ time. Overall this yields a running time of $O(d_v n_v)$. Observe that the size if the surr(v) in the new graph is in $O(d_v n_v)$. Thus, we can compute the planarity region of v by Lemma 6.1 in $O(d_v^2 n_v \log n_v)$ time. □

6.4.2 Finding a Locally Optimal Position

In this section, we are given a vertex v together with its planarity region $\mathcal{PR}(v)$ and we want to compute a locally optimal position. We consider the two cases where v is a pure tail-vertex and the one where v is a pure dummy-vertex. These two cases can be combined to also handle hybrid vertices. For both cases, our approach is the following. For a given angle α, we show how to test whether $\mathcal{PR}(v)$ contains a point with cost less or equal to α. For any $\mathcal{E} > 0$ we can then apply $O(\log(1/\mathcal{E}))$ steps of a binary search over the domain $\alpha \in [0, 2\pi)$ to find a position in $\mathcal{PR}(v)$ whose cost is at most $\mathcal{E}$ larger than the cost of a locally optimal position.

Placing a Pure Tail Vertex.

Let v be a pure tail vertex and let $D(v) \subseteq N(v)$ be the set of dummy neighbors of v, where $N(v)$ is the neighborhood of v; see Figure 6.7. For each dummy neighbor $q \in D(v)$ there is a dissected pair (w_q, q, v) whose angle is active. Note that these are the only active angles of a pure tail vertex. Consider the (oriented) line $\ell(t) = q + t \cdot d_q$ with the direction vector $d_q = q - w_q$. Clearly, placing v onto $\ell(t)$ (for $t > 0$) results in the deviation angle sd-$\alpha(w_q, q, v) = 0$. Moreover, all points in the plane that yield sd-$\alpha(w_q, q, v) \leq \alpha$ lie in a *cone*, i.e., in the intersection (union if $\alpha \geq \pi/2$) of two appropriately chosen half-planes.

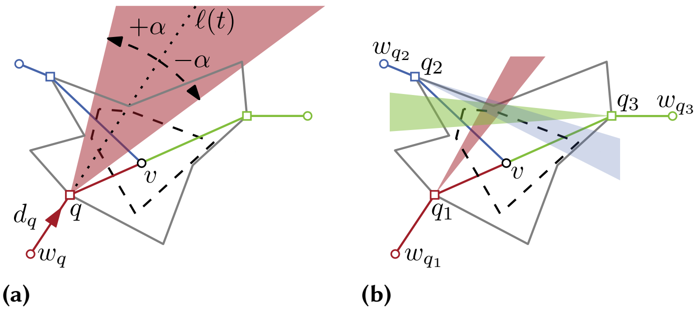


Figure 6.7: (a) A cone with respect to one neighbor q of v. (b) The intersection of all cones with the planarity region (dashed) includes possible positions for the vertex v.

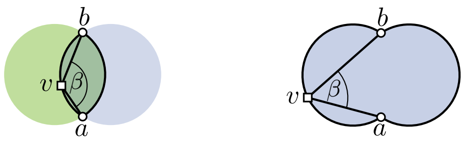


Figure 6.8: The angle $\angle avb$ is at least β for $\beta > 90°$ ($\beta < 90°$) if and only if v lies in the intersection (union) of two discs (including its boundary, but excluding a and b).

It follows, that v can be moved to a position with cost α if and only if the intersection of all cones has a non-empty intersection with the planarity region $\mathcal{PR}(v)$; see for example Figure 6.7. As v has at most d_v dummy neighbors (recall that d_v is the degree of v), the intersections of all cones can be computed in $O(d_v^2 \log d_v)$ time using a sweep-line algorithm [NP82]. Let C be the resulting intersection of the cones. Testing whether C and $\mathcal{PR}(v)$ have non-empty intersection can be done in $O((p_v + d_v^2)\log p_v)$ time, where p_v is the size of $\mathcal{PR}(v)$.

Lemma 6.3. *Let v be a pure tail vertex and assume $\mathcal{PR}(v)$ has already been computed. For any $\epsilon > 0$, an absolute ϵ-approximation of the locally optimal position can be computed in time $O(\log(1/\epsilon)(p_v + d_v^2)\log p_v)$.*

Placing a Pure Dummy Vertex.

A pure dummy vertex v has only two active deviation angles. Let $N(v) = \{a, p, b, q\}$ be the neighbors of v so that (a, v, b) and (p, v, q) are dissected pairs. Consider the angle $\beta = \angle avb$. By a generalization of *Thales' Theorem*, β does not change when moving v on a circular arc with endpoints a and b. Thus, to make sure that β is at least $\pi - \alpha$ (i.e., to ensure that sd-$\alpha(a, v, b) \leq \alpha$), one has to place v in the intersection of two discs (union if $\alpha > \pi/2$); see Figure 6.8. These two disks must have a and b on their boundaries and basic geometry shows that their radii have to be $|ab|/(2\sin(\pi - \alpha))$ (which uniquely defines the two disks).

The same applies for $\angle pvq$. Thus, requiring both active deviation angles sd-$\alpha(a, v, b)$ and sd-$\alpha(p, v, q)$ to be at most α restricts the possible positions of the dummy vertex v either to the intersection of four disks, or to the intersection of the union of two disks with the union of two other disks. The check whether this intersection is empty requires time linear in the size p_v of the planarity region.

Lemma 6.4. *Let v be a pure dummy vertex and assume $\mathcal{PR}(v)$ has already been computed. For any $\epsilon > 0$, an absolute ϵ-approximation of the locally optimal position can be computed in time $O(\log(1/\epsilon) \cdot p_v)$.*

Placing a Hybrid Vertex.

Let v be a dummy vertex with at least one dummy neighbor. Combining the techniques from the two previous sections, we have to check whether $\mathcal{PR}(v)$ has a non-empty intersection with the intersection of up to four cones and up to four disks. This can again be done in time linear in the size p_v of the planarity region. We can thus conclude (for all three types of vertices) with the following theorem.

Theorem 6.5. *Let v be a vertex and assume $\mathcal{PR}(v)$ has already been computed. For any $\epsilon > 0$, an absolute ϵ-approximation of the locally optimal position can be computed in time $O(\log(1/\epsilon)(p_v + d_v^2) \log p_v)$.*

Overall Running Time. We have seen that the planarity region for a vertex v can be computed in $O(d_v^2 n_v \log n_v)$ time (Theorem 6.2) and that a locally optimal position can be approximated in $O(\log(1/\mathcal{E})(n_v + d_v^2) \log p_v)$ time. Note that if v is not a cut-vertex $p_v \in O(n_v)$ otherwise it is in $O(d_v n_v)$. In the following, we assume that $\mathcal{E}$ is a small constant and omit it from the running time.

As the degree d_v of a vertex v is a lower bound for the size n_v of its surrounding, the running time of computing the planarity region dominates the time for computing the locally optimal position. Each iteration thus needs $O(\sum_{v \in V} d_v^2 n_v \log n_v)$ time. In the worst case, this yields the running time stated in the following theorem.

Theorem 6.6. *One iteration of the Geometric Planarization Drawing approach takes $O(n^4 \log n)$ time.*

Observe that since we assume that G has a small number of crossings, a cut-vertex v can not be a dummy vertex; compare Figure 6.9. Thus if consider only biconnected graphs the running time reduces to $O(n^3 \log n)$. The running time improves further to $O(n^2 \log n)$ if the face degrees are bounded by a constant and even to $O(n)$ if additionally the vertex degrees d_v are bounded.

Corollary 6.7. *If G is biconnected, one iteration of the Geometric Planarization Drawing approach takes $O(n^3 \log n)$ time.*

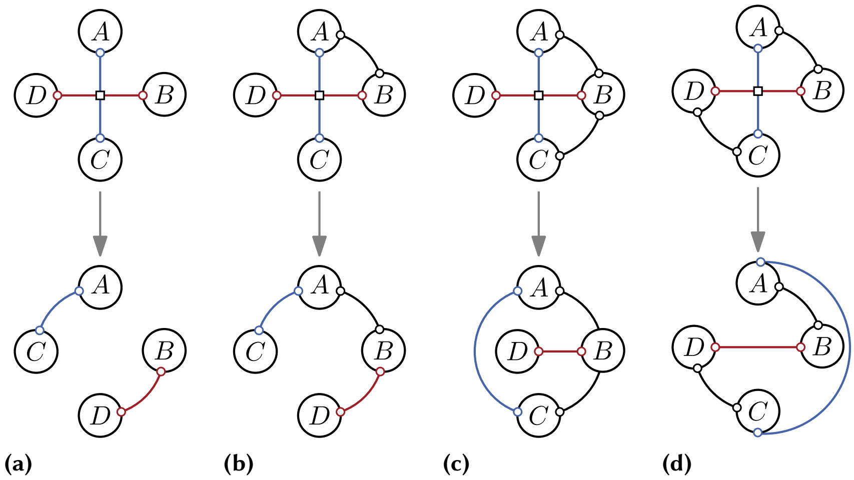


Figure 6.9: Removing a crossing in case that G_p is not biconnected and a dummy vertex is a cut-vertex.

6.5 Evaluation

We present an empirical evaluation of our planarization drawing methods. We first discuss the remaining degrees of freedom in our Geometric Planarization Drawing framework. Afterwards, we describe our experimental setup and the statistical tests we use for the evaluation. The first part of our evaluation focuses on the quality of different configurations of our Geometric Planarization Drawing approach. The second set of experiments focuses on the running time. We evaluate three benchmark sets. We give an extensive evaluation of the *Rome graphs*. Based on the insights obtained from these graphs, we report the results for the *North* and *Community graphs* for a limited number of configurations. We conclude the section with a presentation of a few sample drawings.

6.5.1 Degrees of Freedom in the Geometric Framework

As pointed out above, our algorithmic framework offers quite a number of degrees of freedom and possibilities for tweaking the outcome of the algorithm.

Initial Drawing. We consider two sets of initial drawings ImPrEd and Gc, are both obtained from a planar straight-line drawing computed by PlanarStraightLayout [Kan96] computed with OGDF [Chi+13]. For the first set of initial drawings we applied 100 iterations of ImPrEd, without the forces to optimize the planarization, to the drawings obtained by the PlanarStraightLayout algorithm. For the second

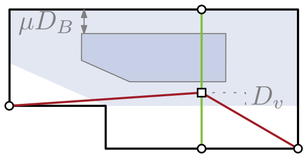


Figure 6.10: Moving the square dummy vertex towards the boundary of the planarity region decreased the deviation angle of the red dissected pair.

set, we iteratively select a vertex and move the vertex to the *geometric center* of its planarity region, i.e., the planarity region is shrunken to a single point. As before, we repeated this process 100 times.

Vertex Orders. We propose different orders for processing the vertices. An Outer Shell is obtained by iteratively removing the vertices of the outer face. An Inner Shell order is the reverse of an Outer Shell, and an Alternating Shell order is obtained by alternating between the two orders. Path Repair is a sequence of vertices where every vertex occurs d_v times. Each edge of the graph G, corresponds to a sequence of vertices of the planarization G_p, namely the vertices on the corresponding planarization path (or an edge) ordered according to their appearance on that path (or the sequence of the two end-vertices if the edge has no crossings). To obtain the Path Repair order, we concatenate these sequences in an order based on a breadth-first search.

Placement of Independent Vertices. For an independent vertex v, every position in the planarity region $\mathcal{PR}(v)$ is equally good since all deviation angles are inactive. To reduce the restrictions imposed by independent vertices on their neighbors, we place v in the geometric center of $\mathcal{PR}(v)$.

Shrinking the Planarity Region. As mentioned before, a locally optimal position for a vertex v might not exists as $\mathcal{PR}(v)$ is an open set; see Figure 6.10. Moreover, it is visually unpleasant when vertices are placed too close to non-incident edges. We thus shrink $\mathcal{PR}(v)$ as follows. Let D_B be the length of the smallest side of the planarity region's bounding box and let $\mu > 0$ be a parameter. Let D_v be the smallest distance from v to a point on the boundary of $\mathcal{PR}(v)$. On one hand, the polygon obtained from shrinking $\mathcal{PR}(v)$ by μD_B may not contain v and therefore can yield a worse deviation angle. On the other hand, if v lies close to the geometric center of $\mathcal{PR}(v)$, shrinking $\mathcal{PR}(v)$ by D_v restricts the movement of v to a small region around v. Hence, we choose to shrink $\mathcal{PR}(v)$ by the minimum of the values μD_B and D_v. In our experiments we used $\mu = 0.1$.

Table 6.1: Configurations for our Geometric Planarization Drawing approach.

Configuration	Vertex Order	Angle Relax. Weight
Alternating Shell	Alternating-Shell	0.0
Shell	Outer-Shell	0.0
Path-Repair	Path-Repair	0.0
Relax-x	Alternating-Shell	$x \cdot 10^{-1}$
		$x \in \{1, 2, 4, 6, 8\}$

Angle Relaxation. While the placement of the tail and hybrid vertices introduced in Section 6.4.2 works independently from the vertex order, it is natural to require that *unplaced* vertices (i.e., vertices that will be moved later in the same iteration) should have a smaller influence on positioning decisions. Hence, we alter the binary search in the cone construction: we replace the opening angle α of the cones of unplaced vertices by $(1-\gamma)\alpha + \gamma\pi$, where $\gamma \in [0, 1]$ is the *angle relaxation weight*, thus widening their cone depending on the value of γ.

Drawing Region. The drawing region is always limited by an axis-aligned rectangle whose side-length is twice as large as the corresponding side-length of the smallest axis-aligned rectangle that entirely contains the initial drawing.

Termination. We consider two possibilities to terminate the execution of our algorithm, (i) after a fixed number of iterations, and (ii) after a fixed period of time. In order to allow a fair comparison between all algorithms in Section 6.5.3, each algorithm gets exactly $5n$ seconds to optimize the drawings. For experiments regarding the running time in Section 6.5.4, we measure the time until convergence limited by 100 iterations.

Configurations. The presented degrees of freedom allow for many different configurations of our algorithm. Table 6.1 lists a set of configurations of our heuristic that we consider in our evaluation. Moreover, we compare these configurations with the baseline algorithm Initial, which simply outputs the initial drawing, and with our modification ImPrEd++ of the force-directed algorithm ImPrEd. The node-node repulsion force and the edge-attraction force used in ImPrEd are parametrized by a value δ. The node-edge repulsion force has a parameter γ. We set both values to $(\log n)^{-1}\sqrt{A/n}$, where A is area of the drawing region and n the number of vertices of the graph. We set the damping factor λ to of the dummy force to 0.1 and the damping factor κ of the force for tail vertices to 0.05.

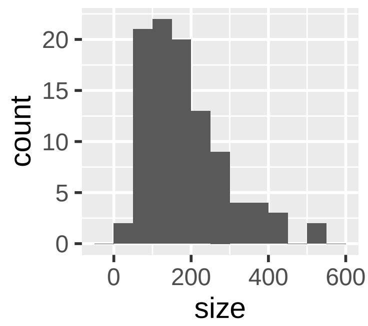


Figure 6.11: The distribution of the *size* of the selected Rome graphs, i.e., the sum of the number of vertices the sum of the number of vertices and edges of a graph.

6.5.2 Experimental Setup and Methodology

We ran the algorithms on 100 randomly selected non-planar Rome graphs[1]. For each of them, we used the largest non-planar biconnected component, Figure 6.11 shows the size distribution of these graphs. To take the lengths of the planarization paths into account, we define three classes of instances: Low ($\mathcal{L}$), Medium ($\mathcal{M}$) and High ($\mathcal{H}$). The partitioning is chosen such that the each class contains a comparable number of graphs. A planarization belongs to $\mathcal{L}$ and to $\mathcal{H}$ if it is at most 2- and at least 6-planar, respectively. Instances in the class $\mathcal{M}$ are k-planar with $2 < k < 6$. There are 33 graphs in $\mathcal{L}$, 40 in $\mathcal{M}$ and 27 in $\mathcal{H}$. The mean number of number of dummy vertices for graphs in $\mathcal{L}$ is 2.7. For the class $\mathcal{M}$ and $\mathcal{H}$ the mean number of dummy vertices is 16.2 and 47.0, respectively.

We applied ImPrEd++ and all configurations of the Geometric Graph Drawing approach listed in Table 6.1 to each graph. In order to be able to apply the binomial test with advantages, see Section 3.2, we have to assign a number to each drawing of a graph. Thus, in the following we consider the *deviation angle* sd-$\alpha(\Gamma)$ *of a drawing* Γ to be the mean of all deviation angles in Γ. Thus, if an algorithm A has an absolute advantage of Δ over an algorithm B on a subset $\mathcal{G}$ of the Rome graphs, this means that the inequality sd-$\alpha(A(G)) + \Delta <$ sd-$\alpha(B(G))$ holds for all graphs $G \in \mathcal{G}'$. For convenience we abbreviate absolute advantages by advantage.

As described in Section 3.2, we randomly partition our benchmark set into a test set $\mathcal{G}_{\text{test}}$ and a verification set $\mathcal{G}_{\text{verify}}$ that each contain 50 graphs. We determine the maximum advantage Δ of an algorithm A over an algorithm B on the set $\mathcal{G}_{\text{test}}$ for a subset of relative size 0.5. We use the set $\mathcal{G}_{\text{verify}}$ to check whether A has a significant advantage of $3/4 \cdot \Delta$ over B for a conjectured probability of 0.5. In this chapter, the conjectured probability is always $p = 0.5$ and thus, we omit this information. Therefore,

[1] `graphdrawing.org/data.html`

the hypothesis that A *has an advantage over* B is short for A *has an advantage over* B *with probability* 0.5.

A δ-*drawing* of a graph G is a drawing of G where each deviation angle is δ. For each algorithm A, we determine the smallest value δ such that A has an advantage of $\Delta = 0°$ over the δ-drawings of the graphs in a subset of $\mathcal{G}_{\text{test}}$. We check on the set $\mathcal{G}_{\text{verify}}$ whether A has significant advantage over the $(4/3 \cdot \delta)$-drawings.

Implementation Details. We use OGDF[2] to planarize the graphs [GMW05] and to compute the initial drawing [Kan96]. We use the libraries CGAL[3] to compute line arrangements, STALGO [HH11, HH12] to shrink polygons, and GMP[4] to represent coordinates.

6.5.3 Quality of the Drawings

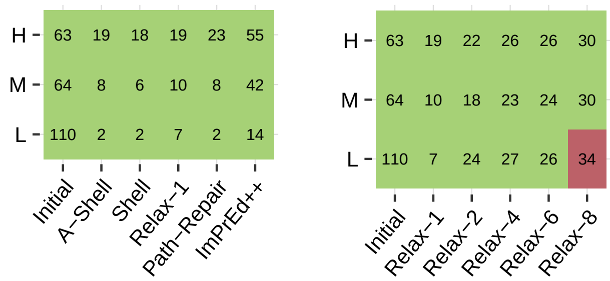
In this Section we discuss the quality of our drawings. The evaluation is guided by the following hypotheses.

I) Gc as an initial drawings yields smaller deviation angles compared to ImPrEd (since the deviation angles of the initial drawings are smaller).
II) The Geometric Planarization Drawing approach and ImPrEd++ each have an advantage over the initial drawing.
III) Geometric Planarization Drawing has an advantage over ImPrEd++.
IV) Relax-1 has an advantage over Relax-2, Relax-4, Relax-6 and Relax-8, respectively.
V) In class $\mathcal{H}$, Relax-1 has an advantage over Alternating-Shell (due to the weakened influence of unplaced vertices).
VI) In the presence of long planarization paths, the Path Repair order has an advantage over other vertex orderings (due to its ability to process all vertices of a planarization path consecutively).

We use Figure 6.12 and Figure 6.13 to show whether the advantages support our hypotheses. The figures are supplemented with the statistics in Table 6.2. A value Δ in a cell in Figure 6.13 is the conjectured advantage of the algorithm A over the algorithm B on the y-axis computed on the training set. Note that the respective maximum advantage on the test set $\mathcal{G}_{\text{test}}$ is $4/3 \cdot \Delta$. A green cell indicates that the advantage is significant, i.e., it is unlikely that the null hypothesis, that for a random Rome graph G the probability of the inequality $\text{sd-}\alpha(A(G)) + \Delta < \text{sd-}\alpha(B(G))$ is at most 0.5, is true

On the contrary, with a red cell we can not reject the null hypothesis. An empty cell, indicates that the algorithm did not have an advantage on the test set. We rounded the values Δ to the largest integer Δ' such that $\Delta' < \Delta$. Therefore, a green cell that

[2] ogdf.net
[3] cgal.org
[4] gmplib.org



(a) Selection **(b)** Relax-x

Figure 6.12: The minimum δ for each configuration (x-axis) such that it has an advantage over a δ-drawing, factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$ (y-axis).

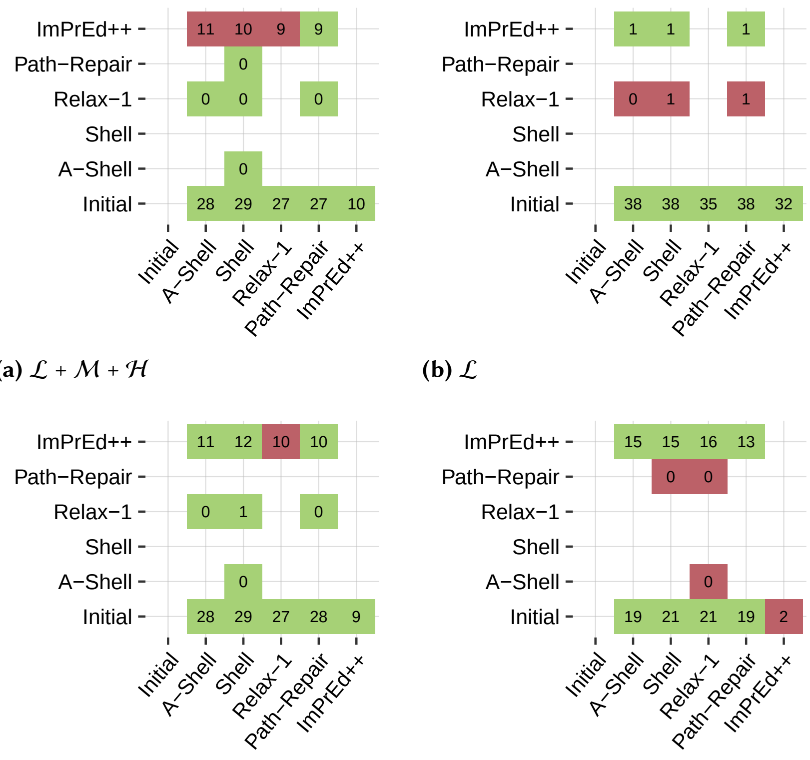
Table 6.2: Median and mean values of the deviation angle for the different algorithms applied to ImPrEd as initial drawing.

	$\mathcal{L}+\mathcal{M}+\mathcal{H}$		$\mathcal{L}$		$\mathcal{M}$		$\mathcal{H}$	
	med	mean	med	mean	med	mean	med	mean
Initial	46.6	51.3	58.3	61.7	45.7	48.2	42.5	43.0
A-Shell	2.29	4.66	0.04	0.33	2.45	3.23	11.6	12.1
Shell	0.78	4.08	0.04	0.04	1.59	2.97	11.3	10.7
Relax-1	6.59	6.56	2.20	3.02	6.61	6.76	10.9	10.6
Relax-2	10.3	9.59	3.21	4.96	10.3	10.6	14.3	13.7
Relax-4	15.3	15.4	12.3	14.5	14.9	15.4	15.8	16.4
Relax-6	16.6	17.2	6.44	16.4	16.7	16.8	18.1	18.6
Relax-8	17.5	19.8	18.0	24.5	15.5	16.8	17.7	18.3
Path Repair	1.81	5.83	0.04	2.21	2.42	4.26	12.7	12.6
ImPrEd++	23.8	20.7	2.78	5.21	25.3	23.7	36.2	35.2

contains a 0 means that the algorithm on the x-axis has advantage of $\Delta < 1$ over the algorithm on the y-axis.

For example, we conjecture, based on the observation in the test set, that the drawings of the Path-Repair configuration have an advantage of 9° over the drawings of ImPrEd++; see Figure 6.13. Thus, there is a subset $\mathcal{G}'$ of $\mathcal{G}_{\text{test}}$ that contains 50% of the graphs of $\mathcal{G}_{\text{test}}$ such that for each graph $G \in \mathcal{G}'$ the inequality sd-$\alpha(\Gamma_1) + 9° <$ sd-$\alpha(\Gamma_2)$, where Γ_1 and Γ_2 are drawings of G computed by Shell and ImPrEd++, respectively. Since the cell is green, the advantage is significant

By Figure 6.12a, for class $\mathcal{L}$ we can say that the deviation angle of drawings computed by the Shell configuration have a significant advantage of over 2°-drawings. This is not necessarily true for $\delta = 1°$. We now discuss our hypotheses.



(a) $\mathcal{L} + \mathcal{M} + \mathcal{H}$

(b) $\mathcal{L}$

(c) $\mathcal{M}$

(d) $\mathcal{H}$

Figure 6.13: Advantage of each configuration (x-axis) compared to each configuration (y-axis), factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$.

Hypothesis I) Good initial drawing. For each configuration, we compared the deviation angles of the final layouts computed by the configuration applied to both sets of initial drawings (Gc and ImPrEd). For each configuration, our test indicated that Gc does not have an advantage over the ImPrEd drawings. Reversely, we were only able to show for the A-Shell configuration and ImPrEd++ that ImPrEd has an advantage of less than 1° over GC. Thus, there is no clear indication that either of the initial drawings results in drawings with smaller deviation angles. In the following, we always use ImPrEd as the initial drawing.

Hypothesis II) Advantage over the Initial drawing. For each configuration, our experiments support this hypothesis, i.e., the advantage over the initial drawing, independent of the configuration, is at least 27°; see Figure 6.13a. Note that the advantage over the Initial drawing decreases with the length of the longest planarization path in a drawing; refer to Figure 6.13b-6.13d. Moreover, Figure 6.13d shows that for the

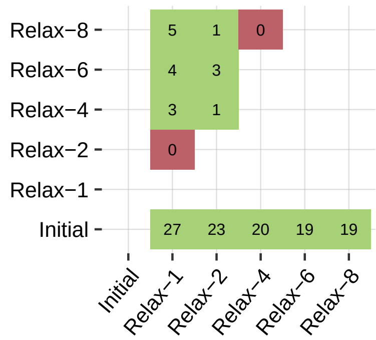


Figure 6.14: Advantages of the Relax-x configurations.

class $\mathcal{H}$ we were not able show that ImPrEd++ has a significant advantage over the Initial drawing.

Hypothesis III) Advantage over ImPrEd++. Figure 6.13a shows that we can only accept the hypothesis for the Relax-1 configuration. For the class $\mathcal{M}$, Figure 6.13c shows that each configuration except of Relax-1 has an advantage of at least 10° over ImPrEd++. For the class $\mathcal{H}$ each configuration has an advantage of at least 15° over ImPrEd++; see Figure 6.13d. Moreover, Figure 6.12a shows that for the class $\mathcal{L}$, ImPrEd has a significant advantage over 14°-drawings, i.e., δ-drawings with $\delta = 14°$. On the other hand, for example, Shell has a significant advantage over 2°-drawings. A similar relation can be observed for the classes $\mathcal{M}$ and $\mathcal{H}$. Overall, we summarize that there are clear indications that the hypothesis is true for graphs with long planarization paths.

Hypothesis IV) Relax-1 has an advantage over Relax-x. Figure 6.14 confirms this hypothesis for $x > 2$. For $x = 2$, we were not able to verify the hypothesis. But note that Relax-1 has a significant advantage over δ-drawings for smaller values of δ compared to Relax-2; see Figure 6.12b. Observe that the statistics listed in Table 6.2 suggest that Relax-8 computes drawings with a smaller deviation angle than Relax-6 for graphs in the class $\mathcal{H}$. The plot in Figure 6.12b on the other hand suggests that the deviation angle of drawings computed by Relax-6 are considerably smaller than the deviation angles of drawings computed by Relax-8.

Hypothesis V) Angle relaxation helps with long planarization paths. The plot in Figure 6.13d does not indicate that there is a significant advantages of Relax-1 over any other configuration of the Geometric Planarization Drawing approach. Moreover, the values for the class $\mathcal{H}$ in Figure 6.12a do not indicate that Relax-1

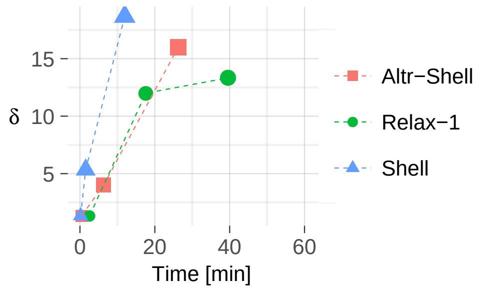


Figure 6.15: Time until convergence versus the δ-value. Symbol sizes indicate the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$. Note: the δ-values of both figures are not coincident due to different experimental setups. The setup for the quality assessment does not allow a running time analysis.

Table 6.3: Mean running time measurements for each configuration.

Configuration	Time per Iteration			Total Time		
	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{H}$	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{H}$
A-Shell	5.2 s	9.7 s	17.3 s	0.4 min	6.3 min	26.3 min
Shell	2.8 s	10.8 s	18.8 s	0.1 min	1.5 min	12.0 min
Relax-1	3.9 s	11.9 s	23.8 s	2.5 min	17.6 min	39.6 min

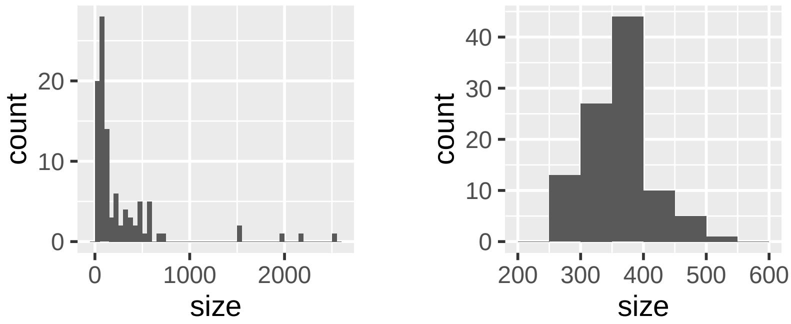
computes drawings with a considerably smaller deviation angle compared to the other configurations. Hence, we conclude that there is no clear support for this hypothesis.

Hypothesis VI) Path Repair helps with long planarization paths. The plot in Figure 6.13d shows that the test on the training set does not conjecture an advantage of the Path Repair configuration over the remaining configurations of the Geometric Planarization Drawing approach. Hence, we do not have any indications that the hypothesis is true.

6.5.4 Running Time

Force-directed methods have been engineered over the past decades. Hence, it is reasonable that the running time of ImPrEd++ is much faster in comparison to our approach that heavily relies on geometric operations. On the other hand, the deviation angle of the drawings obtained by our approach are considerably smaller than the deviation angles of the drawings obtained by ImPrEd++. Therefore, we only evaluate the running time of our Geometric Planarization Drawing approach; see Table 6.3.

Running Time vs. Quality. We use the δ-values to compare the quality of the drawings with respect to the running time. Each point in Figure 6.15 represents final drawings of graph in one of the classes $\mathcal{L}$, $\mathcal{M}$ and $\mathcal{H}$ computed by either the A-Shell, Shell or



(a) North **(b)** Community

Figure 6.16: Size distribution of the North and Community graphs.

Table 6.4: The number of graphs $|\mathcal{G}|$ and the mean number of dummies vertices $\overline{D}$ of graphs in the classes North and Community.

	North		Community	
	$\lvert\mathcal{G}\rvert$	$\overline{D}$	$\lvert\mathcal{G}\rvert$	$\overline{D}$
$\mathcal{L}$	38	1.5	37	28.7
$\mathcal{M}$	33	11.8	25	44.1
$\mathcal{H}$	29	182	38	53.7

Relax-1 configuration of the Geometric Planarization Drawing approach. The figure compares the mean running time required to compute the final drawing against the smallest δ computed with the introduced methodology; all δ-values are confirmed on our verification set. For the class $\mathcal{L}$, all configurations achieve small deviation angles and require on averages less than 2.5 min to compute a drawing. With increasing complexity of the drawings the relevance of the angle relaxation increases. For class $\mathcal{M}$ the Alternating Shell configuration has the smallest δ-value but is slower than the Shell configuration. For drawings of class $\mathcal{H}$, there is no clear dominance. In class $\mathcal{H}$ the Relax-1 configuration yields the best results but the Shell configuration requires less time. We suggest to use the Shell configuration for less complex drawings and when computing time is relevant and for drawings with increasing complexity the Relax-1 configuration.

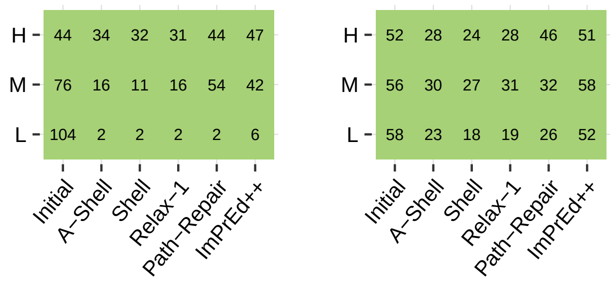
Table 6.5: Median and mean values of the deviation angle for the different algorithms applied to ImPrEd as initial drawing.

	$\mathcal{L}+\mathcal{M}+\mathcal{H}$		$\mathcal{L}$		$\mathcal{M}$		$\mathcal{H}$	
	med	**mean**	**med**	**mean**	**med**	**mean**	**med**	**mean**
North								
Initial	38.4	45.6	48.2	56.7	39.9	45.6	30.8	31.1
A-Shell	4.64	9.10	0.04	0.53	5.87	6.86	22.2	22.9
Shell	0.32	8.22	0.04	0.49	3.13	5.02	22.0	22.0
Relax-1	6.37	9.24	0.04	0.91	7.61	9.13	19.2	20.3
Path Repair	10.2	16.9	0.04	0.59	29.5	23.7	30.8	30.5
ImPrEd++	25.5	19.9	3.07	4.34	26.6	26.0	33.0	33.3
Community								
Initial	37.8	38.3	40.3	40.7	36.4	36.9	37.5	37.0
A-Shell	17.1	16.6	13.0	13.2	17.2	17.4	19.0	19.4
Shell	14.5	14.1	10.6	10.7	14.6	14.6	17.0	17.0
Relax-1	15.0	15.4	12.1	12.4	16.5	16.2	17.6	17.9
Path Repair	19.8	21.7	16.5	16.7	19.4	20.8	24.5	27.2
ImPrEd++	35.4	35.7	37.1	36.2	35.1	34.8	35.4	35.7

6.5.5 North and Community Graphs

In this section we augment our evaluation with a short analysis of two further benchmark datasets. The first dataset contains 100 randomly selected North graphs[5], and the second set contains 100 randomly generated Community graphs, i.e., a set of graphs that resemble Community structure. The community graphs have been used in the evaluation for heuristics to minimize crossings in straight-line drawings of graphs in Chapter 4. Figure 6.16 shows the size distribution of the North and Community graphs. The North graphs are at most 32-planar and the Community graphs are at most 9-planar. For the North graphs the class $\mathcal{L}$ contains only 1-planar graphs, $\mathcal{H}$ contains the graphs that are at least 7-planar, the remaining graphs belong to $\mathcal{M}$. In case of the Community graphs the parameters are selected as follows, $\mathcal{L}$ and $\mathcal{H}$ contains k-planar graphs with $k < 6$ and $k \geq 7$, respectively, and $\mathcal{M}$ contains the remaining graphs. Table 6.4 lists the number of graphs and the mean number of dummy vertices of graphs for both graph classes. We used ImPrEd to compute the initial layout of the North and Community graphs. Moreover, Relax-1 computes drawings with significantly smaller deviation angles than the remaining Relax-x configurations on the Rome graphs. Therefore, we abstain from evaluating the Relax-x configurations

[5] `graphdrawing.org/data.html`



(a) NORTH **(b)** COMMUNITY

Figure 6.17: The minimum δ for each configuration (x-axis) such that it has an advantage over a δ-drawing, factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$ (y-axis).

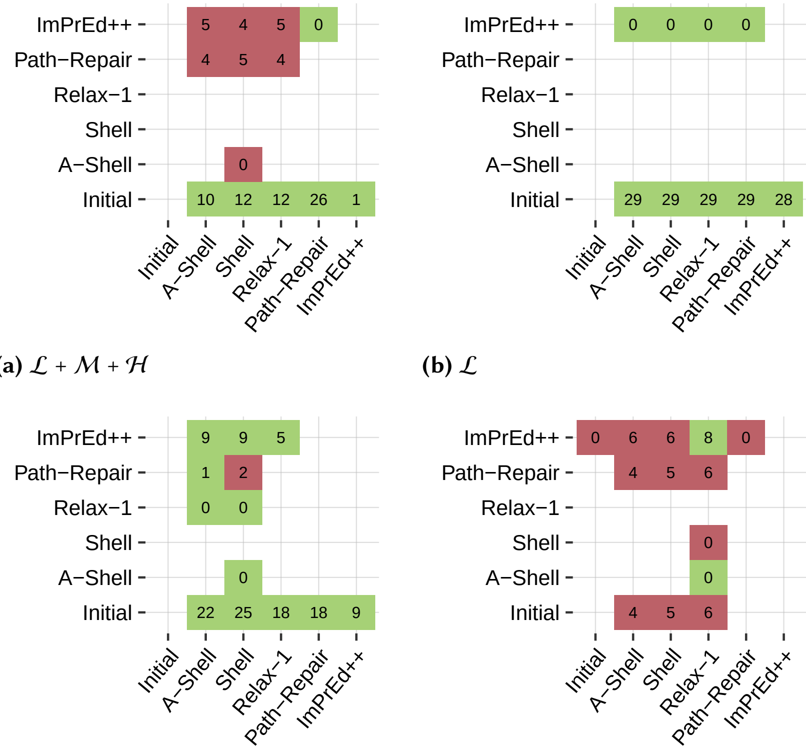
in this section for $x \neq 1$. Table 6.5 lists the mean and median values of the deviation angle of the final drawings of the NORTH and COMMUNITY graphs.

For the COMMUNITY graphs, we can confirm that all heuristics, except IMPRED++, improve the deviation angles; see Figure 6.19. For graphs in the class $\mathcal{L}$ and $\mathcal{M}$ of the NORTH graphs also IMPRED++ improves the deviation angle of the initial drawing significantly. Unfortunately, for the class $\mathcal{H}$ of the NORTH graphs, we were not able to show that any heuristic computes drawings with a smaller deviation angle than another heuristic; refer to Figure 6.18d. Note that this class contains graphs that are k-planar for values in between 7 and 32. On the other hand, all COMMUNITY and ROME graphs at most 13-planar. But observe that the A-SHELL, SHELL and the RELAX-1 configuration have a significant advantages over 34°-drawings. For IMPRED++, we were only able to show that there is a significant advantage over 47°-drawings, indicating that the geometric approach computes drawings with a smaller deviation angle compared to IMPRED++.

Note that similar to the ROME graphs, we can again observe a tendency of the SHELL configuration to compute drawings with the smallest deviation angle. Moreover, there are no clear indications that either the RELAX-1 or the PATH-REPAIR configuration yields drawings with smaller deviation angle of drawings of graphs with long planarization paths, i.e., graphs in the class $\mathcal{H}$.

6.5.6 Sample Drawings

For three graphs Figure 6.20 shows the initial drawing and the drawing after the application of the SHELL configuration of our Geometric Planarization Drawing approach. Observe that the optimization of the tail and dummy vertices in our Geometric Planarization Drawing approach can force a vertex v to be close to an edge which is not incident to v. To increase the readability of the drawings IMPRED can help to

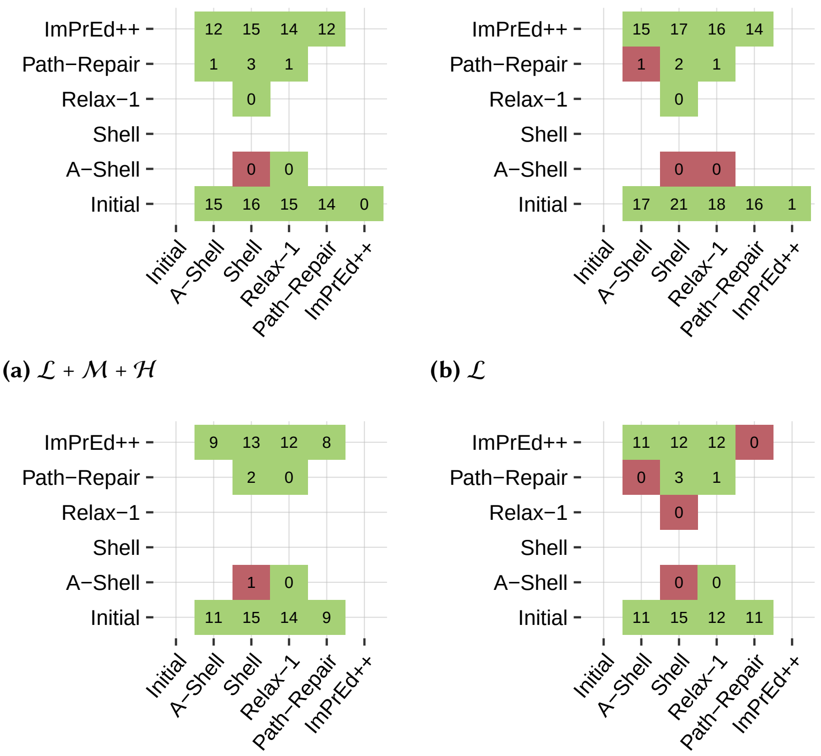


(a) $\mathcal{L} + \mathcal{M} + \mathcal{H}$ **(b)** $\mathcal{L}$

(c) $\mathcal{M}$ **(d)** $\mathcal{H}$

Figure 6.18: North: Advantage of each configuration (x-axis) compared to each configuration (y-axis), factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$.

resolve this issue, i.e., to increase the vertex-edge distances. In case that we want to guarantee, that the deviation angles in the drawings do not change, we apply forces only to independent vertices, i.e., vertices that are neither a dummy or tail vertices. We observed that this strategy can be too restrictive, i.e., the vertex-edge distance remains small, since tail and dummy vertices restrict the movement of the independent vertices. Thus, we propose the following post-processing strategy, that relies on the assumption that ImPrEd, with additional planarization forces, does not alter the deviation angles too much. (i) Replace all planarization paths with an edge from the source to the target vertex if this edge crosses exactly the same edges as the planarization path . (ii) Apply ImPrEd with planarization forces on the remaining dummy and tail vertices on the new drawing. The third column in Figure 6.20 shows examples of drawings that are obtained by this post-processing strategy.

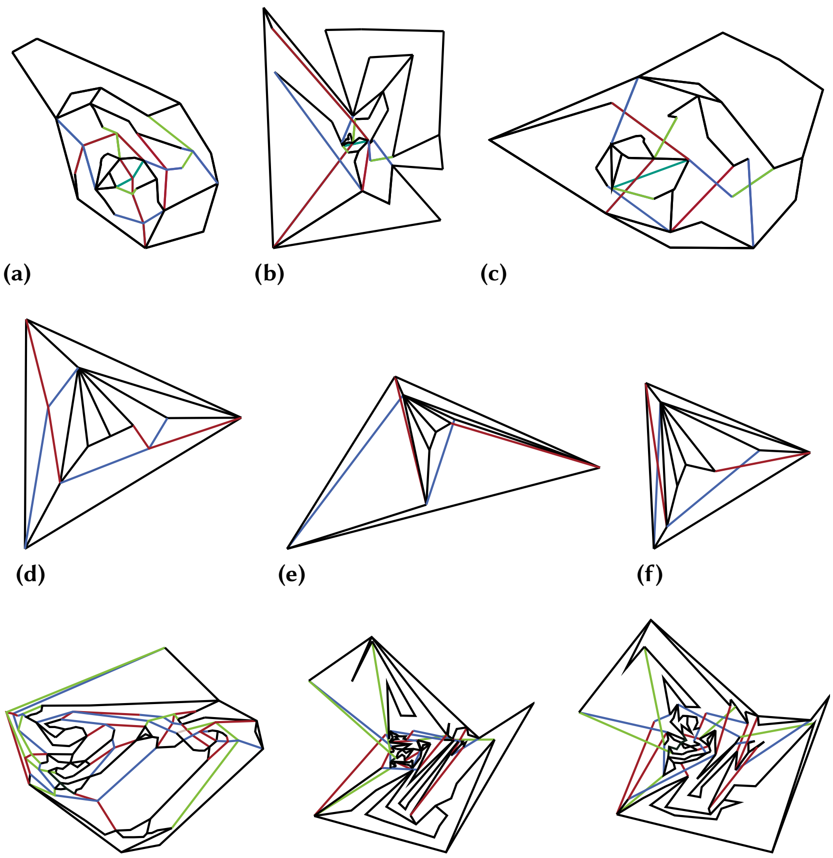


(a) $\mathcal{L} + \mathcal{M} + \mathcal{H}$ (b) $\mathcal{L}$ (c) $\mathcal{M}$ (d) $\mathcal{H}$

Figure 6.19: Community: Advantage of each configuration (x-axis) compared to each configuration (y-axis), factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$.

6.6 Conclusion

We presented two approaches for drawing planarizations such that the edges of the original (non-planar) graph are as straight as possible. Our experiments show that the Geometric Planarization Drawing approach has an significant advantage over our adaptation of the force-directed algorithm ImPrEd, in particular in case of instances with long planarization paths. For instances with short planarization paths, our approach yields drawings that are almost optimal. Even though the deviation angles are worse for instances with longer planarization paths, our Geometric Planarization Drawing approach still significantly improves the deviation angle of the initial drawing. Concerning future research, it would be interesting to see how our geometric approach in Section 6.4 performs when additional optimization criteria such as the angular resolution are incorporated.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 6.20: (a,d,g) Initial drawings, (b,e,h) Final drawing computed with the SHELL configuration, (c,f,i) Drawing with the post processing step. Planarization paths are indicated by colors. (a,b,c) A ROME graph. (d,e,f) A NORTH graph. (g,h,i) A COMMUNITY graph.

7 Crossing-Angle Maximization

The *crossing angle* of a straight-line drawing Γ of a graph $G = (V, E)$ is the smallest angle between two crossing edges in Γ. Deciding whether a graph G has a straight-line drawing with a crossing angle of 90° is $\mathcal{NP}$-hard [ABS12]. We propose a simple heuristic to compute a drawing with a large crossing angle. The heuristic greedily selects the best position for a single vertex in a random set of points. The algorithm is accompanied by a speed-up technique to compute the crossing angle of a straight-line drawing. We show the effectiveness of the heuristic in an extensive empirical evaluation. Our heuristic was clearly the winning algorithm (CoffeeVM) in the Graph Drawing Challenge 2017 [Dev+18].

The chapter is based on joint work with Almut Demel, Dominik Dürrschnabel, Tamara Mchedlidze and Lasse Wulf [Dem+18].

7.1 Introduction

The *crossing angle* cr-$\alpha(\Gamma)$ of a straight-line drawing Γ is defined to be the minimum over all angles created by two crossing edges in Γ. The 24th edition of the annual *Graph Drawing Challenge*, held during the Graph Drawing Symposium, posed the following problem: Given a graph G, compute a straight-line drawing Γ on an integer grid that has a large crossing angle. In this chapter, we present a greedy heuristic that starts with a carefully chosen initial drawing and repeatedly moves a vertex v to a random point p if this increases the crossing angle of Γ. This heuristic was the winning algorithm of the GD Challenge 2017 [Dev+18].

Related Works

A drawing of a graph is called *RAC* if its minimum crossing angle is 90°. Deciding whether a graph has a straight-line RAC drawing is an $\mathcal{NP}$-hard problem [ABS12]. Giacomo et al. [Gia+12] proved that every straight-line drawing of a complete graph with at least 12 vertices has a crossing angle of $\Theta(\pi/n)$. Didimo et al. [DEL11] have shown that every n-vertex graph that admits a straight-line RAC drawing has at most $4n - 10$ edges. This bound is tight, since there is an infinite family of graphs with $4n - 10$ edges that have straight-line RAC drawings. Moreover, they proved that every graph has a RAC drawing with three bends per edge. Arikishu et al. [Ari+12] showed that any n-vertex graph that admits a RAC drawing with one bend or two bends per edge has at most $6.5n$ and $74.2n$ edges, respectively. For an overview over further results on RAC drawings we refer to [DL13a]. Dujmović et al.[Duj+10] introduced the concept of αAC graphs. A graph is *αAC* if it admits a drawing with crossing angle of at least α. For $\alpha > \pi/3$, αAC graphs are *quasiplanar* graphs, i.e., graphs that admit a drawing without three mutually crossing edges, and thus have at most $6.5n - 20$ edges. Moreover, every n-vertex αAC graph with $\alpha \in (0, \pi/2)$ has at most $(\pi/\alpha)(3n-6)$ edges. Besides the theoretical work on this topic, there are a few force-directed approaches that optimize the crossing angle in drawings of arbitrary graphs [ABS13, Hua+10]; see Section 7.2.1.

Contribution

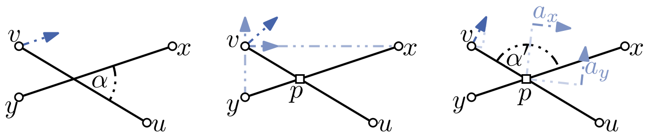
We introduce a heuristic to increase the crossing angle in a given straight-line drawing Γ (Section 7.3). The heuristic is accompanied by a speed-up technique to compute the pair of crossing edges in Γ that create the smallest crossing angle. In Section 7.4 we give an extensive evaluation of our heuristic. The evaluation is driven by three main research questions: i) What is a good parametrization of our heuristic? ii) Does our heuristic improve the crossing angle of a given initial drawing? iii) What is a good choice for an initial drawing?

7.2 Preliminaries

Let Γ be a straight-line drawing of a graph $G = (V, E)$. Denote by n and m the number of vertices and edges of G, respectively. Let e and e' be two distinct edges of G. If e and e' have an interior intersection in Γ, the function $\text{cr-}\alpha(\Gamma, e, e')$ denotes the smallest angle formed by e and e' in Γ. In case that e and e' do not intersect, we define $\text{cr-}\alpha(\Gamma, e, e')$ to be $\pi/2$. The *local crossing angle of a vertex v* is defined as the minimum angle of the edges incident to v, i.e., $\text{cr-}\alpha(\Gamma, v) = \min_{e, uv \in E, e \neq uv} \text{cr-}\alpha(\Gamma, e, uv)$. The *crossing angle* of a drawing Γ is defined as $\text{cr-}\alpha(\Gamma) = \min_{e, e' \in E, e \neq e'} \text{cr-}\alpha(\Gamma, e, e')$. Let Δx and Δy be the difference of the x-coordinates and the y-coordinates of the endpoints of e in a drawing Γ. The *slope of e* is the angle between e and the x-axis, i.e., $\text{slope}(\Gamma, e) = \arctan(\Delta y/\Delta x)$ if $\Delta x \neq 0$ and otherwise $\text{slope}(\Gamma, e) = -\pi/2$.

7.2.1 Force-directed Approaches

In general, force-directed algorithms [Ead84, FR91] compute for each vertex v of a graph $G = (V, E)$ a force F_v. A new drawing Γ' is obtained from a drawing Γ by displacing every vertex v according to the force F_v. Classically, the force F_v is a linear combination of repelling and attracting forces, i.e., all pairs of vertices repel each other, and incident vertices attract each other. It is easy to integrate new forces into this generic system, e.g., in order to increase the crossing angle. For this purpose, Huang et



(a) $F_{\cos}(v)$ **(b)** $F_{\text{cage}}(v)$ **(c)** $F_{\text{ang}}(v)$

Figure 7.1: Sketches of the forces (blue) $F_{\cos}(v)$, $F_{\text{cage}}(v)$ and $F_{\text{ang}}(v)$.

al. [Hua+10] introduced the *cosine force* $F_{\cos}$. The force-directed approach considered by Argyriou et al. [ABS13] uses two forces, F_{cage} and F_{ang}, to increase the crossing angle. In the following we will describe each force.

Let $\overrightarrow{xy}$ denote the unit length vector from x to y. Let uv, xy be two crossing edges in Γ and let α be the angle as depicted in Figure 7.1a and let p denote the intersection point of uv and xy. The cosine force for v is defined as $F_{\cos}(v) = k_{\cos} \cdot \cos\alpha \cdot \overrightarrow{yx}$, where $k_{\cos}$ is a positive constant.

The force $F_{\text{cage}}(v)$ is a compound of two forces $F_{\text{cage}}(v,x)$ and $F_{\text{cage}}(v,y)$; refer to Figure 7.1b. Let l_{ab} denote the distance between two points a and b. Let $l^\star_{vx}$ be the length of the edge vx in a triangle vxp with side length l_{vp} and l_{xp}, and a right angle at the point p. Then, $F_{\text{cage}}(v,x) = k_{\text{cage}} \cdot \log(l_{vx}/l^\star_{vx})\overrightarrow{vx}$, where k_{cage} is positive constant. The force $F_{\text{cage}}(v,y)$ is defined symmetrically.

Again, the force $F_{\text{ang}}(v)$ is a compound of the forces $F_{\text{ang}}(v,x)$ and $F_{\text{ang}}(v,y)$. Consider the unit vector a_x that is perpendicular to the bisector of $\overrightarrow{uv}$ and $\overrightarrow{yx}$; refer Figure 7.1c. Further, let α' be the angle between the $\overrightarrow{uv}$ and $\overrightarrow{yx}$. Then the force $F_{\text{ang}}(v,x)$ is defined as $k_{\text{ang}} \cdot \text{sign}(\alpha' - \pi/2) \cdot |\pi/2 - \alpha'|/\alpha' \cdot a_x$, where k_{ang} is a positive constant. The force $F_{\text{ang}}(v,y)$ is defined correspondingly.

7.3 Multilevel Random Sampling

Our algorithm starts with a drawing Γ of a graph G and iteratively improves the crossing angle of Γ by moving a vertex to a better position, i.e., we locally optimize the crossing angle of the drawing; for pseudocode refer to Algorithm 1. For this purpose, we greedily select a vertex v with a minimal crossing angle cr-$\alpha(\Gamma, v)$. More precisely, let e and e' be two edges with a minimal crossing angle in Γ. We set v randomly to be an endpoint of e and e'. We iteratively improve the crossing angle of v by sampling a set S of T points within a square R and by moving v to the position $p \in S$ that induces a maximal local crossing angle. We repeat this process $L \in \mathbb{N}^+$ times and decrease the size of R in each iteration.

More formally, denote by $\Gamma[v \mapsto p]$ the drawing obtained from Γ by moving v to the point $p = (p_x, p_y) \in \mathbb{R}^2$. Let $R^i(p) = [p_x - s \cdot b^i/2, p_y - s \cdot b^i/2] \times [p_x + s/2, p_y + s \cdot b^i/2] \subset \mathbb{R}^2$

Algorithm 1: Random Sampling

`Input` : Initial drawing Γ, number of levels $L \in \mathbb{N}$, number of samples $T \in \mathbb{N}$, scaling factor $b \in (0,1)$, side length $s > 0$
`Output` : Drawing Γ

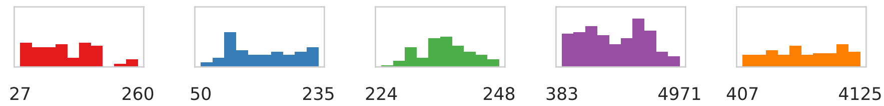
`while` *stopping criteria* `do`
  $(e_1, e_2) \leftarrow$ crossing edges with smallest crossing angle in Γ
  $v \leftarrow$ random vertex in $e_1 \cup e_2$
  `for` $i \leftarrow 1$ `to` L `do`
    $R^i \leftarrow$ square centered at $\Gamma[v]$ with side length $s \cdot b^{i-1}$
    `for` 1 `to` T `do`
      $q \leftarrow$ uniform random position in R^i
      `if` $\text{cr-}\alpha(\Gamma[v \mapsto q], v) > \text{cr-}\alpha(\Gamma, v)$ `then`
        $\Gamma[v] \leftarrow q$

be a square centered at the point p with a *scaling factor* $b \in (0, 1)$ and *initial side length* $s > 0$. Let p^0 be the position of v in Γ and let $S^0 \subset R^0(p^0)$ be a set of T points in $R^0(p^0)$ chosen uniformly at random. Let p^i be a point in $S^{i-1} \cup \{p^{i-1}\}$ that maximizes $\text{cr-}\alpha(\Gamma[v \mapsto p^i], v)$. We obtain a new sample S^i by randomly selecting T points within the square $R^i(p^i)$. Since $\text{cr-}\alpha(\Gamma[v \mapsto p^i], v) = \max_{uv \in E, e \in E \setminus \{uv\}} \text{cr-}\alpha(\Gamma[v \mapsto p^i], uv, e)$, the function can be evaluated in $O(\deg(v)|E|)$ time.

7.3.1 Fast Minimum Angle Computation

The running time of the random sampling approach relies on computing in each iteration a pair of edges creating the minimum crossing angle $\text{cr-}\alpha(\Gamma)$. More formally, we are looking for a pair of distinct edges $e, f \in E$ that have a minimal crossing angle in a straight-line drawing Γ, i.e., $\text{cr-}\alpha(\Gamma, e, f) = \text{cr-}\alpha(\Gamma)$. The well known sweep-line algorithm [BO79] requires $O((n + k)\log(n + k))$ time to report all k intersecting edges in Γ. In general the number of intersecting edges can be $\Omega(m^2)$, but we are only interested in a single pair that forms the minimal crossing angle. Therefore, we propose an algorithm, which uses the slopes of the edges in Γ to rule out pairs of edges, which cannot form the minimum angle.

Assume that we already found two intersecting edges forming a small angle of size $\delta > 0$. We set $t := \lfloor \pi/\delta \rfloor$ and distribute the edges into t buckets $B_0, \ldots, B_{t-1}$ such that bucket B_i contains exactly the edges e with $i\pi/t \le \text{slope}(\Gamma, e) + \pi/2 < (i + 1)\pi/t$. Then each bucket covers an interval of size $\pi/\lfloor \pi/\delta \rfloor \ge \delta$. Thus, if there exist edges e, f with $\text{cr-}\alpha(\Gamma, e, f) < \delta$, they belong to the same or to the adjacent buckets (modulo t). Overall, we consider all pairs of edges in $B_i \cup B_{i+1 \pmod t}$, $i = 1, \ldots t$, and find the pair forming the smallest crossing angle. To find this pair we could apply a sweep-line algorithm to the set $B_i \cup B_{i+1}$. In general this set can contain $\Omega(m)$ edges. Thus, in



(a) North **(b)** Rome **(c)** Community **(d)** 1-Planar **(e)** Tri.+X

Figure 7.2: The distribution of the sum of number of vertices and edges per graph class. The plot is scaled such that a bar of full height would contain 40 graphs.

worst case we would not gain a speed up in comparison to a sweep-line algorithm applied to Γ. On the other hand, in practice we expect the number of edges in a bucket to be small. If we assume this number to be a constant, the overall running time of the exhaustive check is linear in m and does not depend on the number of crossings.

Implementation Details. In the case that the slopes in Γ are uniformly distributed, we expect the number of edges in a bucket to decrease with an decreasing estimate δ. We set the value δ to be the minimal crossing angle of the r longest edges in Γ. In our implementation we set r to be 50 if the graph contains at most 5000 edges, otherwise it is 300.

7.4 Experimental Evaluation

The Random Sampling heuristic has several parameters which allow for many different configurations. In Section 7.4.4, we investigate the influence of the configuration on the crossing angle of the drawing computed by the Random Sampling approach. Further, we address the question whether the Random Sampling approach improves the crossing angle of a given drawing. Our evaluation in Section 7.4.5 answers the question affirmatively. Moreover, we expect that the crossing angle of the drawing computed by the random sampling approach depends on the choice of the initial drawing. We show that this is indeed the case (Section 7.4.6). We close the evaluation with a short running time analysis in Section 7.4.7. Our evaluation is based on a selection of artificial and real world graphs (Section 7.4.1), several choices of the initial drawing; see Section 7.4.2, and a specific way to compare two drawing algorithms; refer to Section 7.4.3.

Setup. All experiments were conducted on a single core of an AMD Opteron Processor 6172 clocked at 2.1 GHz. The server is equipped with 256 GB RAM. All algorithms were compiled with `g++-4.8.5` with optimization mode `-O3`.

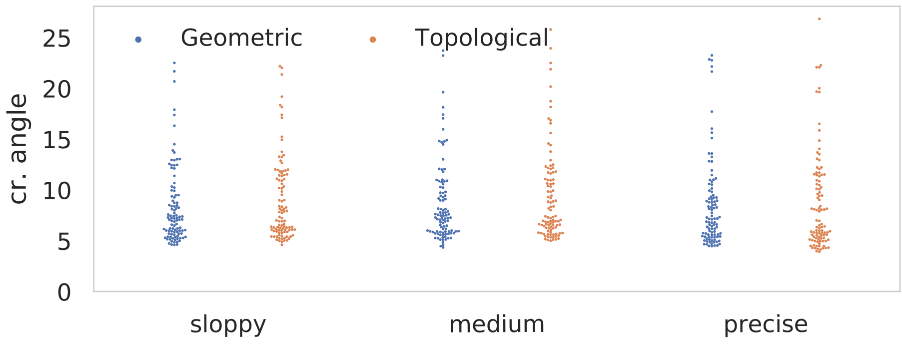


Figure 7.3: The crossing angle in the drawings of the geometric and topological 1-Planar graphs after the application of the Random Sampling approach.

7.4.1 Benchmark Graphs

We evaluate the heuristic on the following graph classes, either purely synthetic or with a structure resembling real-world data. Figure 7.2 shows the size distribution of these graphs. The color of each class is used consistently throughout the chapter.

Real World. The classes *Rome* and *North* (AT&T)[1] are the non-planar subsets of the corresponding well known benchmark sets, respectively. From each graph class we picked 100 graphs uniformly at random. The Community graphs are generated with the LFR-Generator [LFF08] implemented in NetworKit [SSM16]. These graphs resemble social networks with a community structure.

Artificial. For each artificial graph we picked the number n of vertices uniformly at random between 100 and 1000. The Triangulation+X class contains randomly generated n-vertex triangulations with an additional set of x edges. The number x is picked uniformly at random between $0.1n$ and $0.15n$. The endpoints of the additional edge are picked uniformly at random, as well.

The class 1-Planar consists of graphs that admit drawings where every edge has at most one crossing. We used a *geometric* and *topological* procedure to generate these graphs. For the former consider a random point set P of n points. Let $e_1, \ldots, e_k$ be a random permutation of all pairs of points in P. Let $G_0 = (P, \emptyset)$. If the drawing $G_{i-1} + e_i$ induced by P is simple and 1-planar, we define G_i to be this graph, otherwise we set $G_i = G_{i-1}$. We construct the topological 1-Planar graphs based on a random planar triangulation G generated with OGDF [Chi+13]. Let v be a random vertex of G and let v, x, u, y be an arbitrary 4-cycle. We add uv to G if $G + uv$ is 1-planar. The process is repeated x times, for a random number $x \in [0.3n, 0.4n]$.

[1] http://graphdrawing.org/data.html

Table 7.1: Initial drawings with their identifiers used throughout the chapter.

Identifier	Algorithm
Random	uniform random vertex placement
Fr+Cos	FR + Cosine Forces (Section 7.2.1)
Fr+Cage+Ang	FR + Cage + Angular Forces (Section 7.2.1)
Stress	Stress Majorization [GKN05]
Cr-small	Crossing Minimization (Chapter 4)

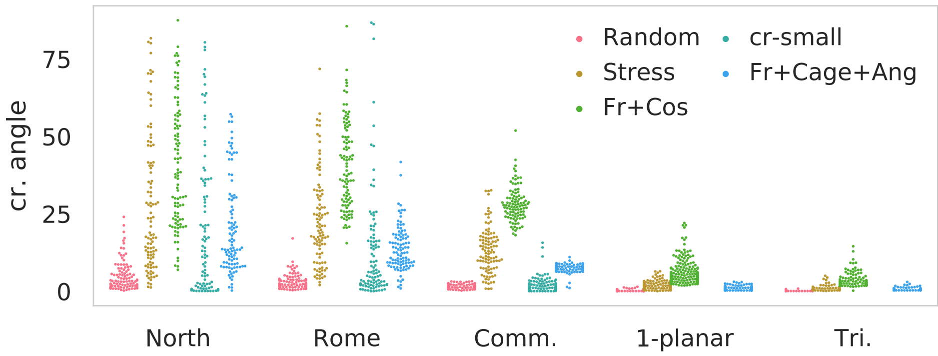


Figure 7.4: Crossing angles of the initial drawings.

Experimental work on the crossing minimization in book embeddings [Jon17] derived different conclusions for the geometric and topological 1-Planar graphs. Figure 7.3 shows the crossing angle for the geometric and topological 1-Planar graphs computed with the random sampling approach. The plot suggests that the distributions do not differ to much. In order to simplify the following evaluation, we merge both graph classes and refer to them as 1-Planar graphs. Thus, the 1-Planar graphs contain 200 graphs where as the other graph classes each consist of 100 graphs.

7.4.2 Initial Drawings

In our evaluation we consider five initial drawings of each benchmark graph; refer to Table 7.1. A random point set P of size n induces a Random drawing of an n-vertex graph. The Fr+Cos drawings are generated by applying our implementation of the force-directed method of Fruchtermann and Reingold [FR91] to the Random drawings with the additional F_{cos} force (Section 7.2.1). The Fr+Cage+Ang drawings are similarly computed as the Fr+Cos drawings, the only difference is that the F_{cos} force is exchanged by the F_{cage} and F_{ang} forces. We applied the stress majorization [GKN05] implementation of the Open Graph Drawing Framework (OGDF) [Chi+13] to Random in order to obtain the Stress drawings. The Cr-small drawings are computed with the heuristic introduced by in Chapter 4 in order to decrease the number of crossings in straight-line drawings. They showed that the heuristic computes drawings with significantly less crossings than drawings computed by stress majorization. Unfortunately, within a feasible amount of time we were not able to compute Cr-small drawings for graphs in the classes 1-Planar and Triangulation+X.

A point in Figure 7.4 corresponds to the crossing angle of an initial drawing. The plot is categorized by graph class. The Random drawings have the smallest crossing angles. The Stress drawings have larger crossing angles than Cr-small and overall, Fr+Cos drawings tend to have the largest crossing angles.

Cage and Angular. Consider the angles in the Fr+Cos and Fr+Cage+Ang drawings. The plot in Figure 7.4. indicates that the cage force produces drawings with the smallest crossing angles. This is not in accord with the claim of Argyriou et al. [ABS13] that they obtained drawings with the largest crossing angle using their implementation of the forces F_{cage} and F_{ang}. Our results are not necessarily comparable, since we may have used different constants to scale the forces. Moreover, we start from different initial drawings. We always start with a random drawing where Argyriou et al. use an organic layout (`SmartOrganic`) provided by yEd[2].

Since our implementation of the force-directed method with F_{cage} and F_{ang} produces drawings with smaller crossing angles than with $F_{\cos}$, we do only consider $F_{\cos}$ in the following evaluation.

7.4.3 Differences between Paired Drawings

In order to compare the performance of two algorithms on multiple graphs and to investigate by how much one of the algorithms outperforms the other, we use concept of advantages introduced Chapter 3. Observe that we introduced this concept context of a minimization problem. In this section we aim for a large crossing angle. Thus, we give a slightly adapted definition of advantages in context of a maximization problem.

We denote by $\Gamma\{G\}$ the set of all drawings of G. Let $\mathcal{G} = \{G_1, G_2, \dots, G_k\}$ be a family of (non-planar) graphs, where $1 \leq k \in \mathbb{N}$. We refer to a set $\Lambda = \{\Gamma_1, \dots, \Gamma_k\}$ as a *family of drawings* of $\mathcal{G}$ where $\Gamma_i \in \Gamma\{G_i\}$. Let Λ^1 and Λ^2 be two families of drawings of $\mathcal{G}$. We say that Λ^1 has an *advantage of* $\Delta > 0$ *on* $\mathcal{F}$ if for all $G_i \in \mathcal{F}$ the inequality $\text{cr-}\alpha(\Gamma_i^1) > \text{cr-}\alpha(\Gamma_i^2) + \Delta$ holds. For a finite set $\mathcal{G}$, we say $\mathcal{F}$ has *relative size at least* $p \in [0, 1]$ if $|\mathcal{F}| \geq p \cdot |\mathcal{G}|$.

In order to compare two families of drawings we plot the advantage as a function of p; refer to Figure 7.6. For each value p the plot contains 5 five bars, each corresponding to a graph class. The height of the bars correspond to advantages Δ for a set of relative size p. A caption of a figure in the form of *A vs B* indicates that if Δ is positive, *B* has

[2] `www.yworks.com`

Table 7.2: Configurations of the RANDOM SAMPLING approach. For each configuration, the scaling factor b is 0.2 and the initial side length s is 10^5.

	Levels L	Sample Size T
SLOPPY	3	50
MEDIUM	4	175
PRECISE	5	400

advantage Δ over A. Correspondingly, if Δ is negative, A has an advantage of $-\Delta$ over B. Thus, Figure 7.6 shows that for $p = 0.1$, for each graph class there is a subset $\mathcal{F}$ of relative size 0.1, i.e., $\mathcal{F}$ contains at least 10 graphs, such that the set SLOPPY has an advantage of Δ over PRECISE on $\mathcal{F}$. In greater detail, SLOPPY has an advantage of 7.9° over PRECISE on the NORTH graphs, 12.9° on the ROME graphs, 11.5° on the COMMUNITY graphs, 1.2° on the 1-PLANAR graphs and 1.2° on the TRIANGULATION+X graphs. On the other side, PRECISE has an advantage of 12.9° over SLOPPY on at least 10 NORTH graphs, 15.7° on the ROME graphs, 13.8° on the COMMUNITY graphs, 1.1° on the 1-PLANAR graphs and 0.4° on the TRIANGULATION+X graphs. Note that only for $p < 0.5$ there can be two disjoint subsets $\mathcal{F}_1, \mathcal{F}_2$ of a graph class of relative size p such that PRECISE has an advantage over SLOPPY on $\mathcal{F}_1$ and SLOPPY has an advantage over PRECISE on $\mathcal{F}_2$.

7.4.4 Parametrization of the Random Sampling Approach

The RANDOM SAMPLING approach introduced in Section 7.3 has four different parameters, the number of levels L, the size of the sample T, the initial side length s and the scaling factor b, that allows for many different configurations. With an increasing number T of samples, we expect to obtain a larger crossing angle in each iteration to the cost of an increasing running time. If we allow each configuration the same running time, it is unclear whether it is beneficial to increase the number of iterations or to increase the number of samples (T) and levels (L) per iteration. This motivates the following question: does the crossing angle of a drawing of an n-vertex graph computed by the random sampling approach within a given time limit t_n increase with an increasing number of samples and levels? We choose to set the time limit t_n to n seconds. This allows for at least $1.6 \cdot n$ iterations for each graph in our benchmark set. Since the parametrization space is infeasibly large, we evaluate three exemplary configurations, SLOPPY, MEDIUM and PRECISE; see Table 7.2.

The plot in Figure 7.5 does not indicate that the distributions of the crossing angle differ across different configurations significantly; further characteristics are listed in Table 7.3. With the plot in Figure 7.6 we can confirm this observation. For each configuration there is only a small subset of each class such that the configuration has

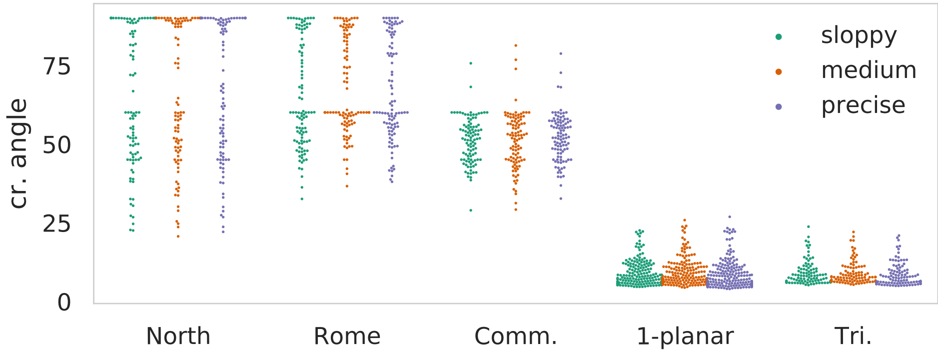
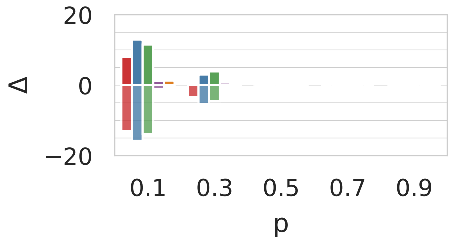
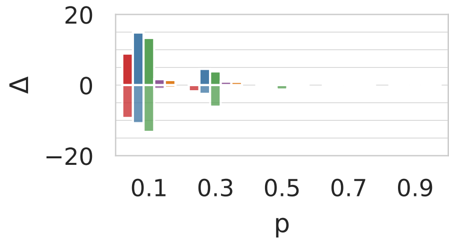


Figure 7.5: Performance of different configurations



(a) Precise vs Sloppy



(b) Precise vs Medium

Figure 7.6: Comparison of the Sloppy configuration to the Medium and Precise configuration. The colors indicate the graph as indicated by Figure 7.2.

an advantage over the other configurations. For example, for the Rome graphs there exist at least 10 graphs such that Sloppy has an advantage of 10° over Precise. On the other hand, there are at least 10 different graphs such Precise has also an advantage of 10° over Sloppy. For $p \geq 0.5$ no configuration has an advantage over the other, or it is negligibly small. Thus, we conclude that given a common time limit, increasing the levels and the sample size does not necessarily increase the crossing angle.

7.4.5 Improvement of the Crossing Angles

In this section, we investigate whether the Random Sampling approach is able to improve the crossing angle of a given drawing within $2n$ iterations. Given the same number of iterations, it is most-likely that we obtain a larger crossing angle of a drawing if we increase the number of samples. Thus, we use the Precise configuration for the evaluation of the above question. We refer to the drawings after the application

Table 7.3: Characteristics computed by different configurations.

graph class	algorithm	crossing resolution			
		min	mean	median	max
Community	Medium	29.13	51.14	52.23	81.27
Community	Precise	32.63	52.01	52.07	78.70
Community	Sloppy	28.90	51.12	51.66	75.61
North	Medium	20.63	67.12	63.87	90.00
North	Precise	22.06	67.82	68.69	90.00
North	Sloppy	22.49	65.84	60.00	90.00
Rome	Medium	36.58	67.85	60.00	90.00
Rome	Precise	37.97	66.43	60.00	90.00
Rome	Sloppy	32.52	64.86	59.98	90.00
1-Planar	Medium	4.33	9.02	7.36	25.79
1-Planar	Precise	3.92	8.60	6.97	26.84
1-Planar	Sloppy	4.58	8.71	7.23	22.50
Triangulation+X	Medium	5.27	8.94	7.66	22.03
Triangulation+X	Precise	4.90	8.55	7.58	20.88
Triangulation+X	Sloppy	5.13	8.90	7.55	23.71

of the Random Sampling approach as Random*, Fr+Cos*, Stress* and Cr-small*, respectively. For characteristics of the crossing angles refer to Table 7.4.

The plots in Figure 7.7 indicate that the Random Sampling approach indeed improves the crossing angle of the initial drawings. Figure 7.8 shows the relationship between the crossing angle of the initial drawing and the final drawing. The plots shows that the Random Sampling approach considerably improves the crossing angle of the initial drawing. In case of the North graphs there are a few graphs that have an improvement of at least 70°. There are at least 10 drawings in Random whose crossing angle is improved by at least 75°; refer to Figure 7.9. For all real world graph classes and all initial layouts there are 70 graphs in each class, such that the final drawing has an advantage of over 25°.

For Triangulation+X, Fr+Cos* has an advantage of at least 11° over Fr+Cos on at least 90 Triangulation+X. For the remaining initial layouts the corresponding advantage is at most 7.6°. Considering the 1-Planar graphs, the corresponding advantages are 14° and 9.7°. This indicates that within $2n$ iterations a large initial crossing angle helps to further improve the crossing angle of 1-Planar and Triangulation+X graphs. Overall, we observe that the 1-Planar and Triangulation+X classes are rather difficult to optimize. This can either be a limitation of our heuristic or the crossing angle of these graphs are indeed small. Unfortunately, we are not aware of meaningful upper and lower bounds on the crossing angle of straight-line drawing of

Table 7.4: Characteristics of the angles in drawings obtained by the Random Sampling approach. Let x and y be the largest and smallest value in a column C for a given graph class. We mark a cell in C with the value v in green, if the $|x - v| \leq 2°$. For the remaining cells, we mark the cell blue, if $|y - v| \leq 2°$.

graph class	layout	crossing resolution			
		min	mean	median	max
Community	Fr+Cos*	49.16	70.63	71.10	88.25
Community	Random*	27.18	37.09	37.24	45.68
Community	Cr-small*	44.09	58.54	58.03	84.61
Community	Stress*	42.89	65.91	63.75	89.09
North	Fr+Cos*	23.82	71.29	78.83	90.00
North	Random*	17.81	55.87	54.51	90.00
North	Cr-small*	18.77	70.55	87.87	90.00
North	Stress*	24.46	70.84	84.68	90.00
Rome	Fr+Cos*	44.52	77.16	81.28	90.00
Rome	Random*	28.14	49.94	47.25	88.43
Rome	Cr-small*	44.19	76.32	84.28	90.00
Rome	Stress*	44.55	77.09	82.70	90.00
1-Planar	Fr+Cos*	13.76	26.55	25.25	53.26
1-Planar	Random*	4.55	6.91	6.02	16.67
1-Planar	Stress*	9.38	15.81	13.85	35.50
Triangulation+X	Fr+Cos*	7.43	18.77	17.24	36.13
Triangulation+X	Random*	4.92	6.79	6.20	15.94
Triangulation+X	Stress*	6.14	11.95	10.41	26.89

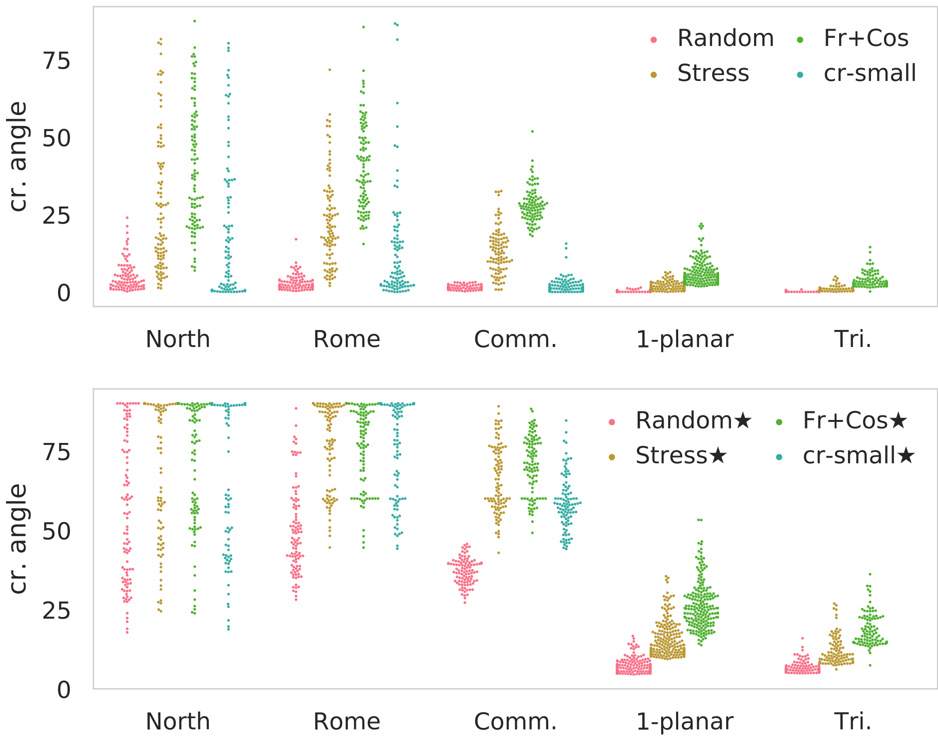


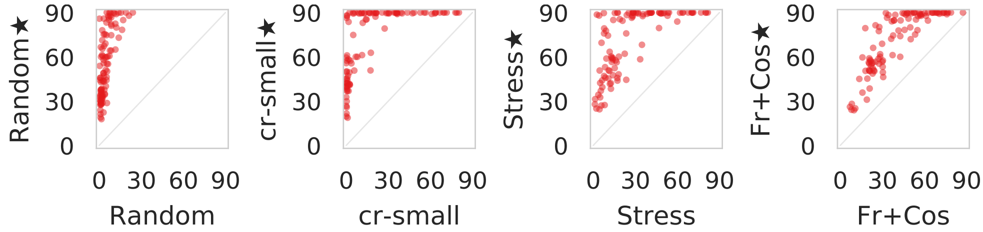
Figure 7.7: Crossing angles before and after applying the Random Sampling approach to the initial drawings.

these graphs. Nevertheless, we can conclude that our heuristic indeed improves the initial crossing angle. To which extend our heuristic is able to increase crossing angle of a drawing depends on the graph class and on the initial drawing itself.

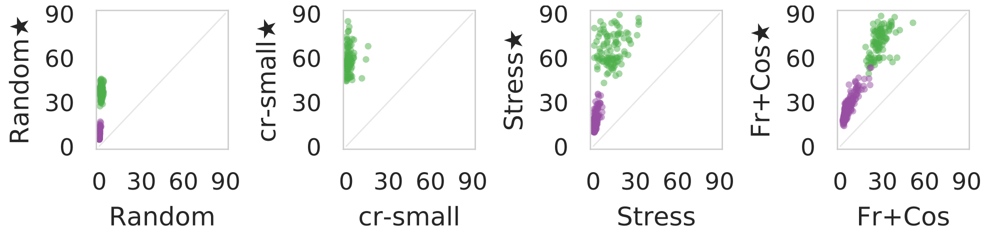
7.4.6 Effect of the Initial Drawing

The Random Sampling approach iteratively improves the crossing angle of a given drawing. Given a different drawing of the same graph the heuristic might be able to compute a drawing with a larger crossing angle. Hence, we investigate whether the choice of the initial drawing influences the crossing angle of a drawing obtained by the Random Sampling approach with $2n$ iterations.

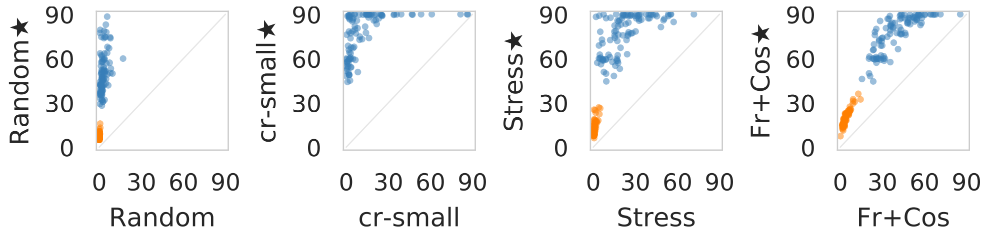
For all graph classes, except from North, it is apparent from Figure 7.7 that the drawings in the set Random* have noticeably smaller crossing angles compared to the remaining drawings. This meets our expectations, since the initial Random drawings presumably have many crossings [HH10] and thus are likely to have many small crossing angles; compare the initial crossing angles plotted in Figure 7.7.



(a) North



(b) Community and 1-Planar

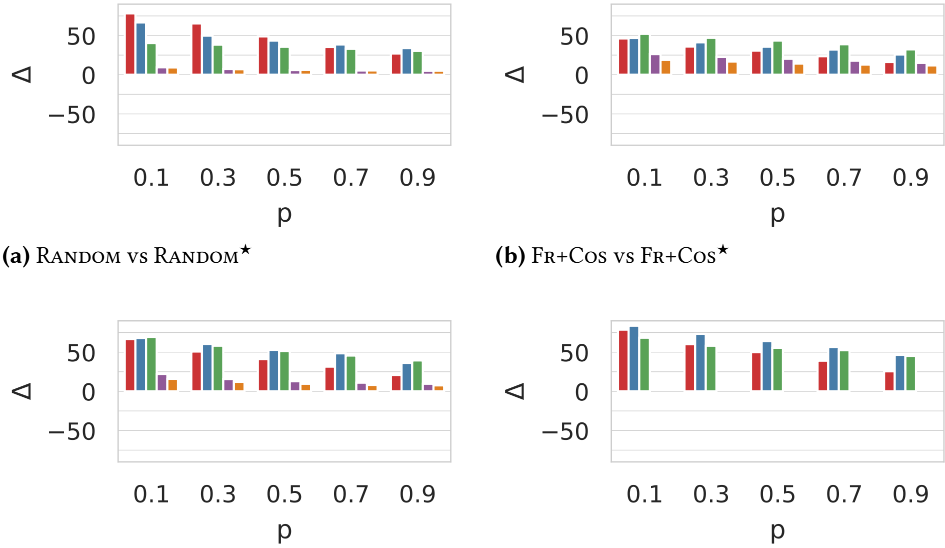


(c) Triangulation+X and Rome

Figure 7.8: Initial crossing angle vs the final crossing angle.

Based on the plot in Figure 7.7 and the characteristics in Table 7.4 we make the following observations. First, on the artificial graph classes (1-Planar and Triangulation+X) and Community, Fr+Cos* contains the drawings with the largest crossing angles. Second, for the real world graph classes (North and Rome) neither Fr+Cos*, Cr-small* nor Stress* clearly contains the drawings with the largest crossing angle. Finally, Random* contains the drawings with the smallest crossing angle, independent of the graph class. In order to corroborate these observations, we use the plots in Figure 7.10 and Figure 7.11

Consider the first claim. The plots in Fig 7.10b and Figure 7.10c clearly show that the observation is true when comparing the Fr+Cos* drawings to Random* drawings. For the comparison of Fr+Cos* to Stress* consider the plots in Figure 7.10h and in Figure 7.10i. For the 1-Planar and Triangulation+X, Fr+Cos* contains the drawings with the largest crossing angles, only with a few exceptions in Triangulation+X.



(a) Random vs Random*

(b) Fr+Cos vs Fr+Cos*

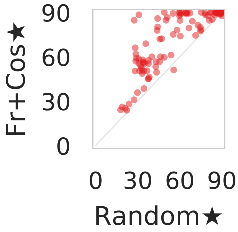
(c) Stress vs Stress*

(d) Cr-small vs Cr-small*

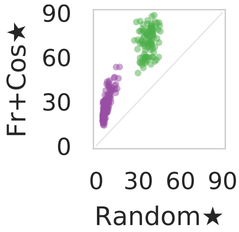
Figure 7.9: Advantages of initial drawings versus the drawings after the application of the Random Sampling approach.

For the Community graphs the plot does not allow for a clear distinction between the two sets of drawings. Indeed, the plot in Figure 7.11a shows that at least 50 Community graphs have drawings in Fr+Cos* with an advantage of 5° over the corresponding drawings in Stress*. When comparing Fr+Cos* to Cr-small* we find that the drawings of Fr+Cos* has an advantage of over 7° over Cr-small* on over 70 Community graphs; see Figure 7.11b. For a subset with at least 10 Community graphs, the advantage rises to almost 25°. We conclude that Fr+Cos* indeed contains the largest crossing angles with respect to graph classes Community, 1-Planar and Triangulation+X.

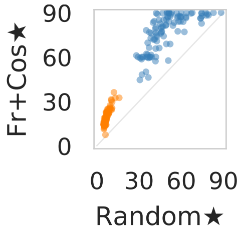
We now turn to the second observation that for the graph classes North and Rome the drawings in Fr+Cos*, Cr-small* and Stress* have comparable crossing angles. For this purpose, consider Figure 7.11a, Figure 7.11b and Figure 7.11d. For all $p \geq 0.3$, there is no set of drawings that has a considerable advantage over another set of drawings. Only for $p = 0.1$, we find that there are 10 North graphs such that Fr+Cos* has an advantage of at least 5° over Stress*. Vice versa there are 10 different North graph such that Stress* has an advantage of at least 5° degrees over Fr+Cos*. The comparison on the Rome graphs yields similar results. Thus, we conclude that there is no considerable difference between the Fr+Cos* and Stress* drawings. Based on the



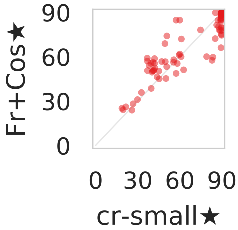
(a) RANDOM



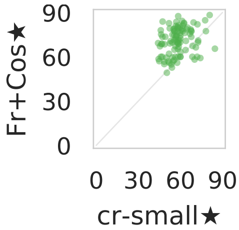
(b) RANDOM



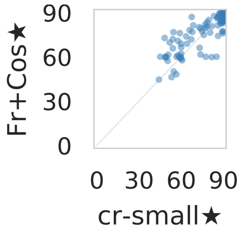
(c) RANDOM



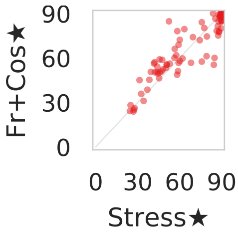
(d) CR-SMALL



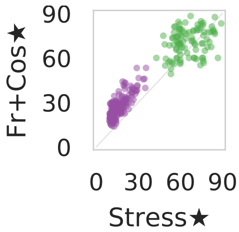
(e) CR-SMALL



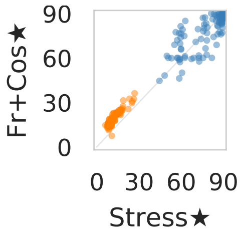
(f) CR-SMALL



(g) STRESS



(h) STRESS



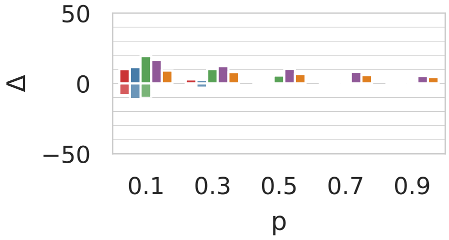
(i) STRESS

Figure 7.10: Comparison of the initial layout.

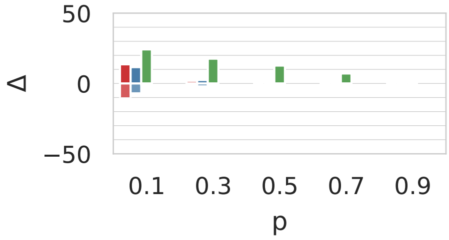
plots in Figure 7.11b and Figure 7.11d we draw the same conclusion for the comparison of FR+COS* to CR-SMALL* and STRESS* to CR-SMALL*.

Based on Figure 7.7 we already observed that the RANDOM* drawings contains drawings the smallest crossing angles. Only for the NORTH class, the plot is not conclusive. The plot in Figure 7.11c shows that there are at least 70 graph such that FR+COS* has an advantage of 4.5° over RANDOM*. For $p = 0.5$ the advantage increases to over 14°.

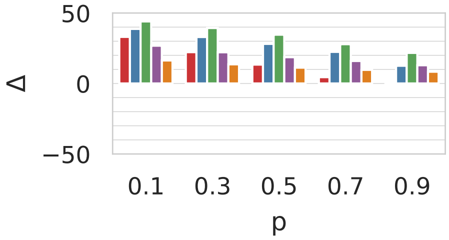
Overall, we conclude that the RANDOM SAMPLING approach computes the largest crossing angle when applied to the FR+COS drawings, in particular for the artificial graph classes. This is plausible, since the crossing angles of the initial crossing angles are already good. As shown in the previous section, depending on the graph class,



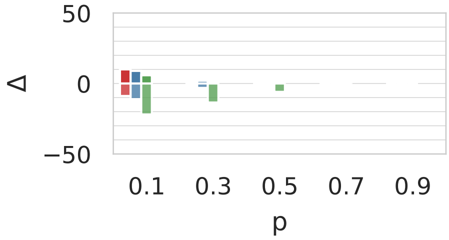
(a) Stress★ vs Fr+Cos★



(b) Cr-small★ vs Fr+Cos★



(c) Random★ vs Fr+Cos★



(d) Stress★ vs Cr-small★

Figure 7.11: Advantages of the Random Sampling approach applied to different initial drawings.

there is a large improvement in the crossing angle, if we start with such an initial drawing.

Increasing the Number of Iterations. Since the initial crossing angles in Fr+Cos are larger in comparison to Stress, we investigate in this section, whether we are able to decrease the advantage of Fr+Cos★ over Stress★ if we increase the number of iterations for Stress★. For this purpose, we applied to $4n$ iterations to the initial drawings in Stress. We refer to this drawings as Stress★★ and compare them to Fr+Cos★; see Figure 7.12 and Figure 7.12c. We observe that at least 50 of North and Rome graph have drawings in Stress★★ with a slightly larger crossing angles than in the corresponding drawing Fr+Cos★. On the other hand, Fr+Cos★ has an even larger advantage over Stress★★ on the remaining graph classes. Thus, indicating that on the artificial graph classes Fr+Cos indeed is a good choice for an initial drawing for the Random Sampling approach.

7.4.7 Note on the Running Time

In this section we shortly evaluate the running time of our algorithm on all our graphs. First, we report the running time of the three configurations Sloppy,Medium and

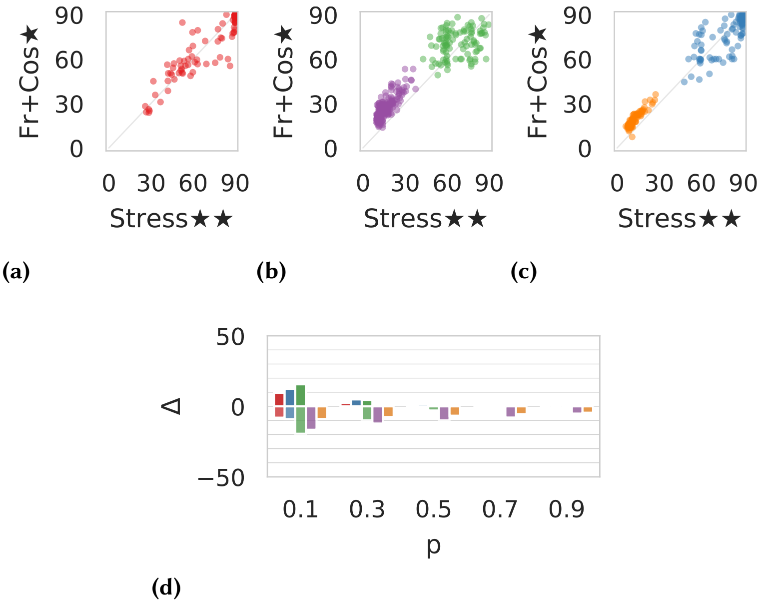
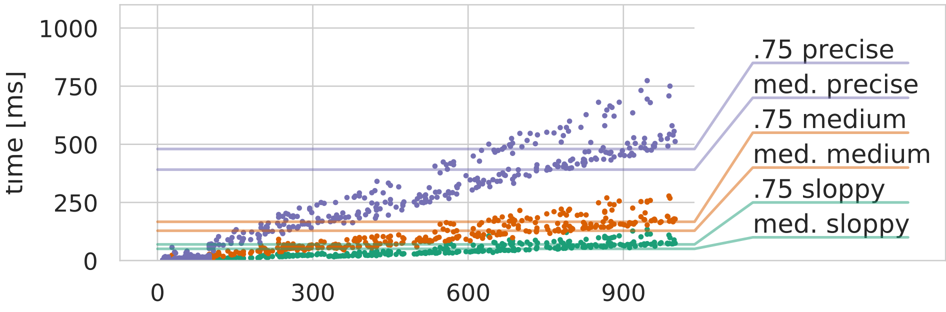


Figure 7.12: Comparison of $4n$ random sampling steps in Stress** to $2n$ steps in Fr+Cos*.

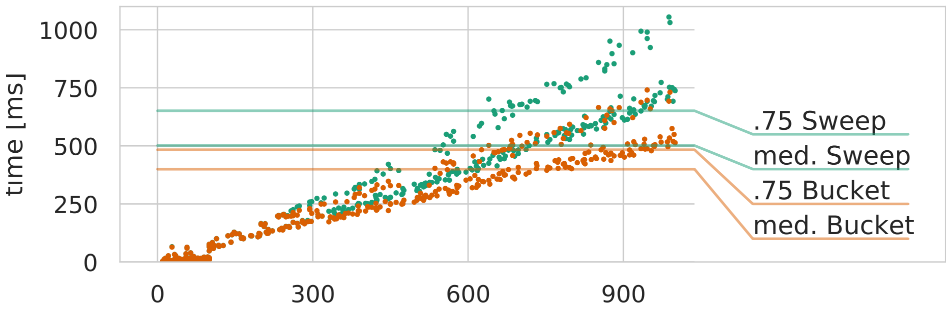
Precise. Second, we show that the heuristic introduced in Section 7.3.1 improves the average running time.

The plot in Figure 7.13a contains a point (n, t) for each graph G in our benchmark set, where n denotes the number of vertices of G and t is the average running time for a single iteration, i.e., the average time to move a single vertex. We used Random as initial drawings. The color indicates the configuration and the lines show the median and the 0.75-percentile over the set of all running time measurements with respect so one configuration. As expected, the plots shows a clear order of the configurations: Sloppy needs less time than Medium, Medium requires less time than Precise.

We now compare the running time of the speed-up technique introduced in Section 7.3.1 to a sweep-line approach. For this purpose, we applied two implementations of the Random Sampling heuristic with the Sloppy configuration to the Random drawings. The Sweep implementation uses a sweep-line algorithm to compute the pair of crossing edges that create the smallest crossing. Bucket uses the algorithm described in Section 7.3.1. We employ the speed-up technique only for graphs with at least 1000 edges, we refer to these graphs as *large*. Fig 7.13b plots the running time per iteration for n-vertex graphs. The plot in Figure 7.13b uses the same convention as we used in Figure 7.13a. Bucket has an average running time of 391 ms per iteration on the large graphs and Sweep has an average running time of 500 ms. On all graphs. Bucket requires on average 328 ms per iteration.



(a) Running for the tested configurations of the RANDOM SAMPLING approach.



(b) Sweep Line Algorithms vs Bucket

Figure 7.13: Average running time per iteration. *med.* corresponds to the average running time overall instances. The *.75* corresponds to 0.75-percentile.

Note that this time comparison heavily depends on our implementation of SWEEP and BUCKET. To show that for large graphs BUCKET uses less operations, we compare the number of crossings in a drawing Γ to the number of edge-pairs that BUCKET compares to find the smallest angle in Γ. Note that the number of crossings is a lower bound for the number of operations of any implementation of the SWEEP approach.

In more detail, a data point (P, C) in Figure 7.14 corresponds to a single graph. We counted in each iteration the number of crossings C_i of the current drawing and the number of tested edge-pairs P_i. The values C and P correspond to the average of these values over $2n$ iterations, i.e., $(P, C) = (\sum_i P_i/(2n), \sum_i C_i/(2n))$. Note that the plot uses a double log-scale. Ideally, P is significantly smaller than C. We observe that for small instances, the heuristics tests more edge-pairs than the drawing has crossings. With an increasing number of crossings, the heuristic indeed tests less edge-pairs than crossings.

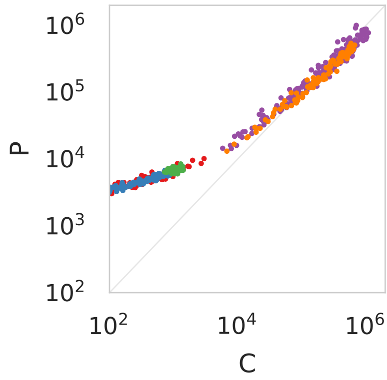
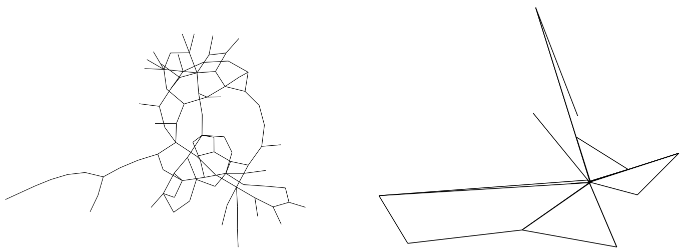


Figure 7.14: Number C of crossings vs number P of tested edge-pairs.

7.5 Conclusion

We designed and evaluated a simple heuristic to increase the crossing angle in a straight-line drawing of a graph. On our benchmark set our heuristic computes drawings with larger crossing angles compared to our implementation of force-directed approaches that are designed to optimize the crossing angle. Further, our evaluation shows that the performance of our heuristic depends on the initial drawing and on the graph class. In particular, on real world networks our heuristic is able to compute larger crossing angles than on artificial networks. This can either be a limitation of our heuristic or it can be a property of the artificial graph classes, i.e., the crossing angle of any drawing of a graph in an artificial graph class is small. We are not aware of lower and upper bounds of the crossing angle of these graphs. Thus, investigating such bounds of the 1-Planar and Triangulation+X graphs is an interesting theoretical question.



(a) **(b)**

Figure 7.15: (a) Stress drawing of a Rome graph. (b) Drawing after optimizing the crossing angle. The ratio between the longest and shortest edge is large.

Figure 7.15 shows that our heuristic does not necessarily compute readable drawings. Nevertheless, parts of the Random Sampling heuristic are easily exchangeable. For example, the objective function can be replaced by a linear combination of number of crossing and the crossing angle, or the sampling region R_i can be replaced by an arbitrary polygon in order to preserve some properties of the drawings, e.g., the number of crossings. Thus, future work can be concerned with adapting the Random Sampling approach with the aim to compute readable drawings.

Part II

Stretching Topological Embeddings with Constraints

8 Inserting an Edge into a Geometric Embedding

The algorithm to insert an edge e in linear time into a planar graph G with a minimal number of crossings on e [GMW05], is a helpful tool for designing heuristics that minimize edge crossings in drawings of general graphs. Unfortunately, some graphs do not have a geometric embedding Γ such that $\Gamma + e$ has the same number of crossings as the embedding $G + e$. This motivates the study of the computational complexity of the following problem: Given a combinatorially embedded graph G and a face f, compute a geometric embedding Γ with outer face f that has the same combinatorial embedding as G and that minimizes the crossings of $\Gamma + e$. Eades et al. [Ead+15] characterized the embeddings of G and e that are stretchable when the choice of the face f is free. In this chapter, we characterize the stretchable embeddings when f is given as part of the input, thereby we answer an open question of Eades et al. Moreover, we give polynomial-time algorithms for special cases and prove that the general problem is fixed-parameter tractable in the number of crossings. Moreover, we show how to approximate the number of crossings by a factor $(\Delta - 2)$, where Δ is the maximum vertex degree of G.

This chapter is based on joint work with Iganz Rutter [RR18].

8.1 Introduction

Crossing minimization is an important task for the construction of readable drawings. The problem of minimizing the number of crossings in a given graph is a well-known $\mathcal{NP}$-complete problem [GJ83]. A very successful heuristic for minimizing the number of crossings in a topological drawing of a graph G is to start with a spanning planar subgraph H of G and to iteratively *insert* the remaining edges into a drawing of H. The edge insertion problem for a planar graph G and two vertices $s, t \in V(G)$ asks to find a drawing $\Gamma + st$ of $G + st$ with the minimum number of crossings such that the induced drawing Γ of G is planar. The problem comes with several variants depending on whether the drawing Γ can be chosen arbitrarily or is fixed [GKM08, GMW05]. In the planar topological case both problems can be solved in linear time. More general problems such as inserting several edges simultaneously [CH16] or inserting a vertex together with all its incident edges [Chi+09] have also been studied.

All these approaches have in common that they focus on topological drawings where edges are represented as arbitrary curves between their endpoints. By contrast, we focus on geometric embeddings, i.e., planar straight-line drawings, and the corresponding rectilinear crossing number. In this scenario we are only aware of a few heuristics that compute straight-line drawings of general graphs; compare Chapter 4, Chapter 5 and [Kob13]. Clearly, if a geometric embedding Γ of the input graph G is provided as part of the input, there is no choice left; we can simply insert the straight-line segment from s to t into the drawing and count the number of crossings it produces. If, however, only the combinatorial embedding and possibly the outer face is specified, but one may still choose the vertex positions so that this results in a geometric drawing with the given combinatorial embedding, then the problem becomes interesting and non-trivial. We call this problem *geometric edge insertion*.

Contribution and Outline.

In Section 8.2, we recite the characterization of *stretchable edges* given by Eades et al. [Ead+15], in case of a combinatorially embedded graph with the choice of the outer face. They left the characterization of stretchable edges in combinatorially embedded graphs with a fixed outer face as an open question. We connect this problem to the stretchability of a single pseudoline in a topologically embedded graph and provide a characterization of edges stretchable with respect to a fixed outer face.

In Section 8.3 we consider the complexity of the geometric edge insertion problem for combinatorially embedded graphs of bounded degree where the choice of the outer face is free. Namely, we give a quadratic-time algorithm for the case that the maximum degree Δ of G is at most 5. For the general case, we give a $(\Delta - 2)$-approximation that runs in linear time. In Section 8.4 we consider combinatorially embedded graphs with a fixed outer face. We give an efficient algorithm for testing in special cases whether there exists a way to insert the edge st so that it does not produce more crossings than when we allow to draw it as an arbitrary curve. Finally, we give a randomized FPT algorithm that tests in $O(4^k n)$ time whether an edge can be inserted with at most k crossings (Section 8.5).

8.2 Characterization

The aim of this section is to characterize embeddings of planar graphs and edges st that allow for a straight-line drawing $\Gamma + st$ with a minimal number of crossings. Let $G = (V, E)$ be a planar graph with a given combinatorial embedding where only the choice of the outer face is free. Additionally, let s and t be two distinct vertices with $st \notin E$. Denote by $G + st$ the graph G together with the edge st. We want to insert the *edge st into the embedded graph G*. That is, we seek a straight-line drawing Γ of G (with the given embedding) such that st can be inserted into Γ with a minimum number of crossings. In Γ, the edge st starts at s, traverses a set of faces and ends in t. Topologically, this corresponds to a path $p(\Gamma)$ from s to t in the *extended dual $G^\star_{st}$ of G*,

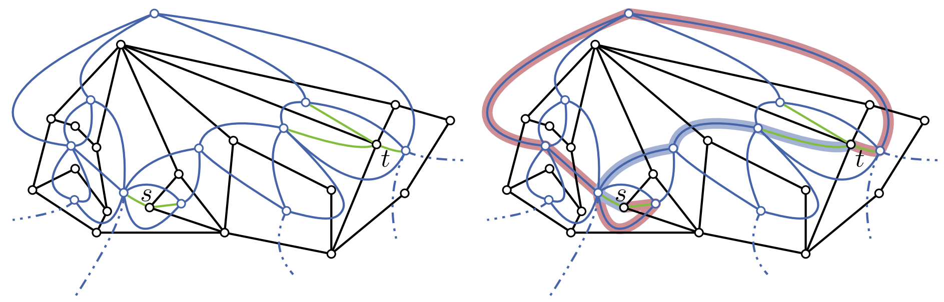


Figure 8.1: (a) The extended dual (green + blue) of the primal graph (black) and the vertices corresponding to s and t. (b) An st-path p and a p-friendly st-path p'. The union of both induces a pseudoline with respect to the embedded primal graph.

i.e., in the dual graph $G^\star$ plus s and t connected to all vertices of their dual faces; see Figure 8.1a. An extended dual $G^\star_{st}$ has a topological drawing $\Gamma^\star_{st}$ so that each primal edge crosses its dual edge exactly once in $\Gamma + \Gamma^\star_{st}$, the position of s and t in Γ and in $\Gamma^\star_{st}$ coincide, and the edge incident to s and t are crossing-free; compare Figure 8.1. The number of crossings in $\Gamma + st$ corresponds to the length of the path $p(\Gamma)$ minus two. However, not all st-paths in $G^\star_{st}$ are of the form $p(\Gamma)$ for a straight-line drawing Γ of G. If an st-path p is in the form $p(\Gamma)$, we say that p is *stretchable*. In case that we fix the choice of the outer face o and p has the form $p(\Gamma)$ for a straight-line drawing Γ with this particular outer face, we say that p is *stretchable with respect to o*.

A labeling of G is a mapping $l\colon V \longrightarrow \{L, R\}$ that labels vertices as either left or right. Consider an edge uv of G that is crossed by a path p such that u and v are to the left and to the right of p, respectively. The edge uv is *compatible* with a labeling l if $l(u) = L$ and $l(v) = R$. A path p of $G^\star_{st}$ and a labeling l of G are *compatible* if l is compatible with each edge that is crossed by p. A path p is *consistent* if there is a labeling of G that is compatible with p. Eades et al. [Ead+15] show the following result.

Proposition 8.1 (Eades et al. [Ead+15], Theorem 1)**.** *An st-path in $G^\star_{st}$ is stretchable if and only if it is consistent.*

Eades et al. ask for a characterization of the edge insertion problem in case of combinatorially embedded graphs with a fixed outer face. For this purpose, consider a topologically embedded planar graph $G = (V, E)$ in the Euclidian plane. A (oriented) closed Jordan curve $\mathcal{L}$ is a *pseudoline with respect to G* if it passes through the outer face and for each edge e of G, $\mathcal{L}$ either entirely contains e or crosses e at most once.

Theorem 8.2 (Da Lozzo et al. [Da +18], Theorem 10.16 in Chapter 10)**.** *For every pair of a planar embedded graph G and a pseudoline, there is a straight-line drawing Γ of G and an oriented line L with the following properties:*

1. *Γ has the same combinatorial embedding and outer face as G,*
2. *vertices of G to the left of $\mathcal{L}$ are to the left of L in Γ,*
3. *vertices of G to the right of $\mathcal{L}$ are to the right of L in Γ,*
4. *L intersects in Γ the same vertices and edges as $\mathcal{L}$ in G, and*
5. *they do so in the same order.*

Before we characterize the stretchable st-paths, we introduce some notations. Two paths p and p' are *edge-disjoint* if they do not share an edge. Two paths p and p' of an embedded graph are *non-crossing* if at each common vertex v, the edges of p and p' incident to v do not alternate in the cyclic order around v in the graph induced by p and p'. Let f be a face of G and the corresponding dual vertex as $f^\star$. Let p be an st-path. An st-path p' is *(p, f)-friendly* if (i) p and p' are edge-disjoint, (ii) p and p' are non-crossing, and (iii) p' contains $f^\star$. An st-path is *p-friendly* if it is (p, o)-friendly for the outer face o of an embedded graph G. The definition of p-friendly paths and Theorem 8.2 yields the following characterization of stretchable st-paths.

Theorem 8.3. *For a combinatorial embedded graph G and a face f, an st-path p in the extended dual $G^\star_{st}$ is stretchable with respect to f if and only if there is a (p, f)-friendly st-path p' in $G^\star_{st}$.*

Proof. We first show that, if there is a (p, f)-friendly path p' that p is stretchable with respect to f. Let Γ be a topological drawing of G with f as the outer face and let $\Gamma^\star_{st}$ be the corresponding drawing of $G^\star_{st}$. Then, the paths p and p' in the drawing of $\Gamma^\star_{st}$ define a curve ρ; compare Figure 8.1. Since each edge of G crosses in $\Gamma \cup \Gamma^\star_{st}$ its dual edge exactly once, the curve ρ intersects each edge of G in Γ at most once and s and t are the only two vertices of G that are on ρ. Since p' is (p, f)-friendly, ρ passes through f which is the outer face of Γ. The paths p and p' are only edge-disjoint. Therefore, p and p' can share a set of vertices and ρ may self-intersect. Since p and p' are non-crossing, we can perturb ρ at each intersection point to resolve these intersections. Therefore, ρ is a pseudoline with respect to Γ. By Theorem 8.2 there is a straight-line drawing of G with outer face f and a line L such that s and t lie on L and the segment st intersects the same edges as p and st does so the same order.

Conversely, assume that p is stretchable with respect to f. Then by Theorem 8.2 there is a straight-line drawing of G with outer face f such that the segment st intersects the same edges as p and in the same order; see Figure 8.2. Let g be the line that contains the segment st. Each edge of G intersects g at most once. Thus, the complement of st in g defines a path from s to t in $G^\star_{st}$ that is edge-disjoint from p, does not cross p and that contains f. Hence, we find a (p, f)-friendly st-path p'. ◻

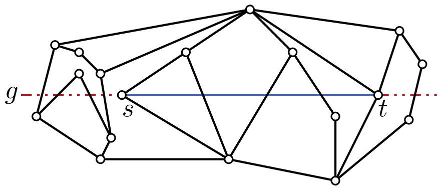


Figure 8.2: The line g through the segment st induces a path in the extended dual.

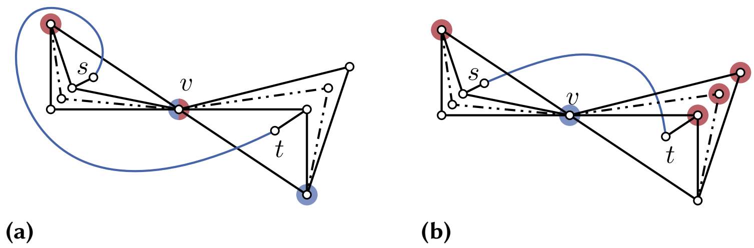


Figure 8.3: Ratio between length of the shortest st path and the length of a shortest consistent st-path. The solid black edges induce a graph of maximum degree 6. Red vertices have label L, blue vertices have label R. (a) The shortest path from s to t in $G^\star_{st}$ is not consistent.

For an embedded graph with a fixed outer face immediately have the following corollary.

Corollary 8.4. *For an embedded graph G with the outer face o, an st-path p in the extended dual $G^\star_{st}$ is stretchable with respect to o if and only if there is a p-friendly st-path p' in $G^\star_{st}$.*

Overall, depending on the setting we now have the following combinatorial tools to compute straight-line drawings $\Gamma + st$ with a minimal number of crossings. If the choice of the outer face is free, we look for a consistent st-path of minimum length, i.e., for an appropriate $\{L, R\}$-labeling of the vertices. If the outer face is fixed, we look for two edge-disjoint paths p and p', while minimizing the length of p and requiring that p' is p-friendly, i.e., it does not cross p and contains the vertex dual to the outer face of G.

8.3 Bounded Degree

In this section, we study consistent st-paths in combinatorially embedded planar graphs of bounded degree, i.e., the choice of the outer face is free. The graph in Figure 8.3a shows that there is a graph with maximum vertex-degree 6 where every shortest st-path is not consistent. In particular, Figure 8.3b indicates that every consistent st-path has a linear number of crossings. This motivates the question whether shortest

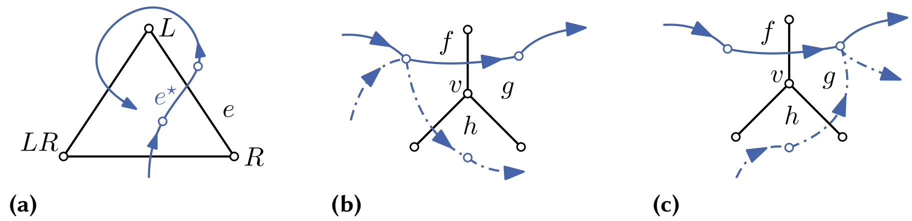


Figure 8.4: (a) Labeling induced by the blue path. An inconsistent path around a degree 3 vertex either visits face f (b) or face g (c) twice.

st-paths in graphs with a smaller bounded degree are consistent. We show that every shortest st-path in planar graphs of bounded degree 3 is consistent, and that in each planar graph with vertex degree at most 5, there is a shortest st-path that is consistent. Finally, we prove that there is a consistent st-path of length $(\Delta - 2) \cdot l$ in a graph with maximum vertex degree Δ and a shortest st-path of length l in $G^\star_{st}$.

Let p be an st-path in $G^\star_{st}$ and let $e^\star$ be an edge of p. An endpoint u of the primal edge e of $e^\star$ is *left of* $e^\star$ if it is locally left of p on e; refer Figure 8.4a. A vertex v of G is *left* (*right*) of p if v is left (right) of an edge of p. We now consider a labeling extended by two more labels LR, $\perp$. We define the labeling l_p induced by p as follows. Each vertex that is left and right of p gets the label LR. The remaining vertices that are either left or right of p get labels L and R, respectively. Vertices neither left nor right of p get the label $\perp$. Obviously, there is a labeling l of G compatible with p if and only if l_p does not use the label LR.

Theorem 8.5. *Let G be a planar embedded graph of degree at most 3. Then every shortest st-path in $G^\star_{st}$ is consistent.*

Proof. Let p be a shortest path in $G^\star_{st}$. Assume that p is not consistent. Then there is a vertex v that is left and right of p. Let fg be the first edge of p that crosses a primal edge incident to v. If the degree of v is at most 2, then p contains either a loop or a double edge, contradicting the assumption that p is a shortest path. Therefore, assume that the degree of v is 3. Without loss of generality, let f, g and h be the faces around v in clockwise order; see Figure 8.4b and Figure 8.4c. Since v is left and right of p, p contains either the edge fh or hg. Thus, p contains either f or g twice. This contradicts the assumption that p is a shortest path. □

In the following, we denote by $p[v_i, v_j]$, with $1 \le i \le j \le k$, the subpath of a path p from v_i to v_j, i.e., $p[v_i, v_j] = \langle v_i, v_{i+1}, \dots, v_j \rangle$.

Theorem 8.6. *Let G be a planar embedded graph with maximum degree 5. Then there is a shortest st-path in $G^\star_{st}$ that is consistent.*

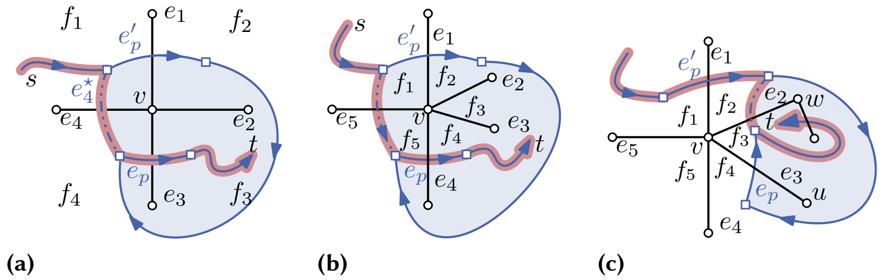


Figure 8.5: Inconsistent path around (a) a degree-4 vertex and (b,c) a degree-5 vertex. The shaded region depicts the region ρ.

Proof. Let p be a shortest st-path in $G^\star_{st}$. We call an edge e_p of p *good* if the vertices left and right of it do not have label LR in the labeling l_p induced by p. We say a suffix $p[x,t]$ of p, with $x \in V(p)$, is *good* if all its edges are good.

Our proof strategy is to incrementally increase the length of the longest suffix of p that is good. If p is not consistent, then let e_p denote the last edge of p that is not good. Then an endpoint v of the primal edge corresponding to e_p has label LR. Without loss of generality, we may assume that v lies left of e_p. Since $l_p(v) = LR$, there is an edge e_p' of p that has v to its right. By the choice of e_p, it follows that e_p' lies before e_p on p. We now distinguish cases based on the degree of v.

Case 1, deg(v) ≤ 3: If $\deg(v) \leq 3$, then we find as in Theorem 8.5 that p enters or leaves a face twice, which contradicts the assumption that it is a shortest st-path.

Case 2, deg(v) = 4: If $\deg(v) = 4$, we denote the edges around v in clockwise order as $e_1, \ldots, e_4$ such that e_p' crosses e_1, i.e., $e_p' = e_1^\star$. Moreover, we denote the faces incident to v in clockwise order as $f_1, \ldots, f_4$ where f_1 is the starting face of e_p'; see Figure 8.5a.

Since $e_p' = f_1f_2$ and no face has two incoming or two outgoing edges of p, it follows that $e_p = f_4f_3$ crosses e_3. Since p is a shortest path, it follows that $f_2 = f_4$, i.e., $p[f_1, f_2] = p[f_1, f_4] = \langle f_1, f_2\rangle$. Let p' be the path obtained from p by replacing the subpath $p[f_1, f_4]$ by the edge $e_4^\star$ that crosses e_4, i.e., $e_4^\star = f_1f_4$; see the thick red path in Figure 8.5a. By construction, it is $l_{p'}(v) = L$. Observe that $p'[f_4, t] = p[f_4, t]$ lies inside the region ρ bounded by $p[f_1, f_4]$ and a curve connecting f_1 and f_4 that crosses e_4. The only vertex inside this region whose label changed is v. Therefore, the path $p'[f_1, t]$ consists of good edges, and we have thus increased the length of the suffix of the shortest path that consists of good edges.

Case 3, deg(v) = 5: Now assume that $\deg(v) = 5$. We denote the edges around v as $e_1, \ldots, e_5$ in clockwise order such that e_p' crosses e_1. We further denote the faces incident to v in clockwise order as $f_1, \ldots, f_5$ such that e_p' starts in f_1. Since no face

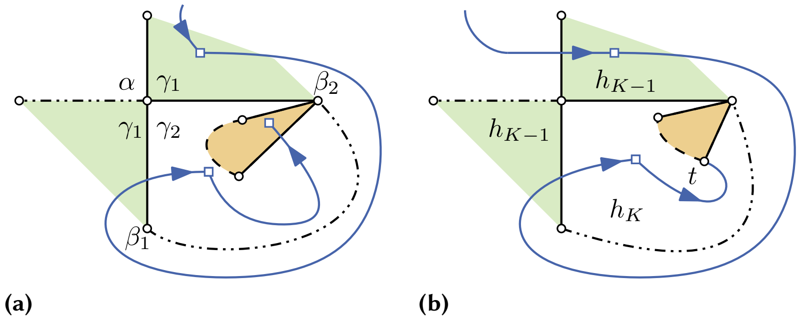


Figure 8.6: (a) A problematic face γ_1. (b) The faces h_{K-1} and h_K are not problematic.

has two incoming or two outgoing edges, it follows that either e_p crosses e_4 from f_5 to f_4 (Figure 8.5b) or e_p crosses e_3 from f_4 to f_3 (Figure 8.5c).

If e_p crosses e_4, then we obtain p' by replacing $p[f_1, f_5]$ by the single edge that crosses e_5; see thick red path in Figure 8.5b. As in the degree-4 case, we find that $f_2 = f_5$ and v is a cutvertex and that $p'[f_1, t]$ consists of good edges. Therefore, in case that e_p crosses e_4, we increased the length of the longest suffix consisting of good edges.

We now consider the case that e_p crosses e_3. Consider the path q obtained from p by replacing the subpath $p[f_2, f_3]$ with the edge $e_2^\star$ that crosses e_2, i.e., $e_2^\star = f_2f_3$; see Figure 8.5c. Let $e_2 = vw$. Since w and t lie in the same region bounded by $p[f_2, f_4]$ and f_2f_3, the suffix $q[f_2, t]$ is good if only if $l_p(w) = \perp$ or $l_p(w) = L$. Thus, before we construct the path p' from p, we first rebuild p to a path that satisfies this condition.

Claim 7. *There is a shortest st-path p'' such that p'' crosses e_3, $l_{p''[f_2,t]}(w) = \perp$ or $l_{p''[f_2,t]}(w) = L$, and the suffix $p''[f_2, t]$ is good.*

Assume that the claim is true. Then, first, let p'' with the properties of the claim. Afterwards, we obtain the path p' from p'' as described by replacing $p''[f_2, f_3]$ by the edge $e_2^\star$. Then, in all cases, we increase the length of the suffix of the shortest path consisting of good edges. Eventually, we thus arrive at a shortest path whose edges are all good and that hence is consistent. Thus, to finish the proof of the theorem, it is only left to prove the claim.

Proof of the Claim. Observe that, if $l_{p[f_2,t]}(w) = \perp$ or $l_{p[f_2,t]}(w) = L$, p already meets the requirements for p'', i.e., we set $p'' := p$. Thus, we restrict our attention to the case that $l_{p[f_2,t]}(u) = l_{p[f_2,t]}(w) = R$, where u is the endpoint of e_3 distinct from v. We first give a formal definition of the structure that forbids us to reroute p via the edge $e_2^\star$ dual to vw. Afterwards, we locally modify p in order to reduce the number of problematic cases.

Let γ_1, γ_2 be two distinct faces that share two edges $\alpha\beta_1, \alpha\beta_2$ on their boundary and such that $p[\gamma_1, t] = \gamma_1 \cdot p[\gamma_2, t]$ and p crosses $\alpha\beta_1$; see Figure 8.6a. We say the face γ_1 is *problematic with respect to* p if $l_{p[\gamma_1,t]}(\beta_1) = l_{p[\gamma_1,t]}(\beta_2)$.

Recall that $f_2 = f_4$ and that $p[f_2, t] = f_2 \cdot p[f_3, t]$, where the edge f_2f_3 is dual to e_3. Recall that $e_2 = vw$ and $e_3 = vu$. If f_2 is not problematic, then by definition $l_{p[f_2,t]}(u) \neq l_{p[f_2,t]}(w)$. Hence, p is already the desired path.

Assume that f_2 is problematic with respect to p. Let $h_1, h_2, \ldots, h_K$ be the faces of G that are crossed by $p[f_2, t]$ in this order, i.e., $h_1 = f_2 = f_4, h_2 = f_3$. Note that t lies on the boundary of h_K and that p does not cross an edge incident to t. Therefore, h_K and h_{K-1} are not problematic with respect to p; compare Figure 8.6b. Since f_2 is problematic, there is maximum number k, for $1 \leq j < K - 1$, such that the first k faces $h_1, h_2, \ldots, h_k$ are problematic with respect to p but h_{k+1} is not problematic. Initially, let $p_{k+1} = p$. In the following we describe how to obtain an st-path p_j from p_{j+1} while ensuring the following invariants:

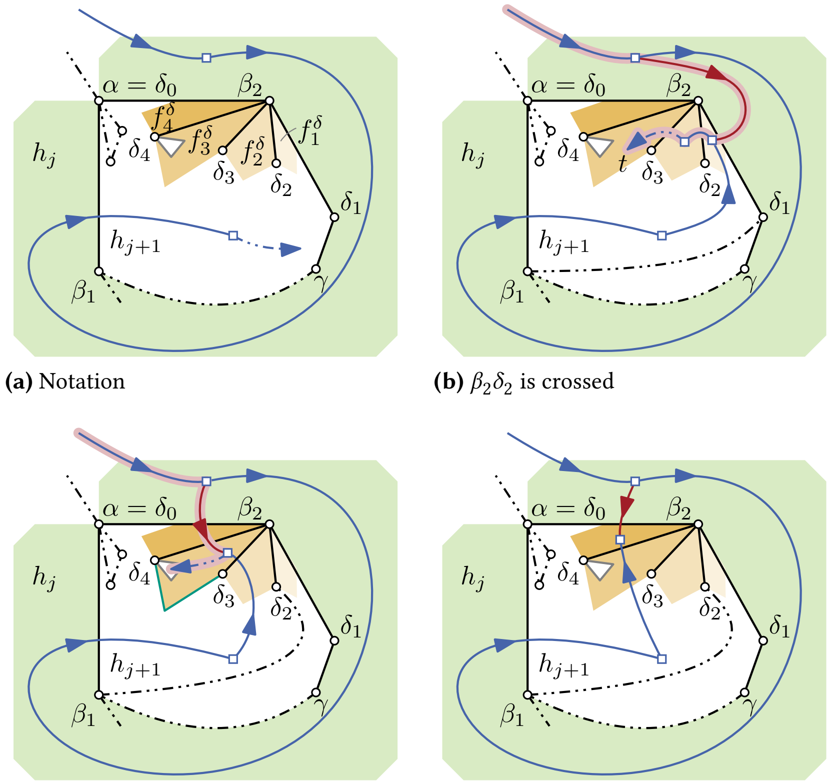
(i) p_j is a shortest st-path,
(ii) $p_j[s, h_j] = p_{j+1}[s, h_j]$,
(iii) $p_j[h_j, t]$ is good, and
(iv) h_j is not problematic with respect to p_j.

Thus, we eventually arrive in the case that $h_1 = f_2$ is not problematic with respect to p_0 and $p_0[f_2, t]$ is good. Hence, we set $p'' = p_0$ to prove the claim. Thus, we now consider the case that h_j is problematic and h_{j+1} is not problematic with respect to p_{j+1}.

Let $\alpha\beta_1, \alpha\beta_2$ be the edges incident to h_j and h_{j+1} such that $\alpha\beta_1$ is crossed by $p_{j+1}[h_j, t]$; see Figure 8.7a. Let $\delta_0, \delta_1, \ldots, \delta_d$, with $d < 5$, be the neighbors of β_2 in clockwise order, where $\delta_0 = \alpha$. Moreover, denote by f_i^δ the face that contains $\beta_2\delta_i$ and $\beta_2\delta_{i+1}$ on its boundary, where we set $d + 1 := 0$. Observe that $f_0^\delta = h_j$. Moreover, since h_j is problematic, i.e., $\beta_2\delta_0$ (and $\beta_1\delta_0$) are on the boundary of h_{j+1}, it follows that $f_d^\delta = h_{j+1}$. Without loss of generality, assume that β_1 lies to the right of the path $p_{j+1}[h_j, t]$, i.e., $l_{p_{j+1}[h_j,t]}(\beta_1) = R$. Since h_j is problematic it follows that β_2 has label R, i.e., $l_{p_{j+1}[h_j,t]}(\beta_2) = R$. Thus, there is an edge $\beta_2\delta_i$ that is crossed by $p_{j+1}[h_j, t]$. Observe that since p_{j+1} is a shortest path, this edge can neither be $\beta\delta_0$ nor $\beta_2\delta_1$ as otherwise $p_{j+1}[h_k, t]$ would visit h_k twice. We distinguish cases based on whether $\beta_2\delta_2$, $\beta_2\delta_3$ or $\beta_2\delta_d$ is crossed by $p_{j+1}[h_k, t]$; see Figure 8.7.

Case 3.1, $\beta_2\delta_d$ is crossed: As observed before, we have that $f_d^\delta = h_{j+1}$. Then, the edge dual to $\beta_2\delta_0$ is a short cut for the path $p_{j+1}[h_{j+1}, t]$; compare Figure 8.7d. A contradiction to the assumption that p_{j+1} is a shortest path.

Case 3.2, $\beta_2\delta_2$ is crossed: We obtain p_j from p_{j+1} by replacing the path $p_{j+1}[h_j, f_1^\delta]$ by the edge dual to $\beta_2\delta_1$; see thick red path in Figure 8.7b. Since $p_{j+1}[h_j, f_1^\delta]$ is a shortest-path that contains the edge dual to $\alpha\beta_1$, it follows that p_j has the same length as p_{j+1}. By construction, we have that $p_j[s, h_j] = p_{j+1}[s, h_j]$. Note that $p_j[h_{j+1}, t] = p_{j+1}[h_{j+1}, t]$



(a) Notation

(b) $\beta_2\delta_2$ is crossed

(c) $\beta_2\delta_3$ is crossed

(d) $\beta_2\delta_d$ is crossed

Figure 8.7: Case distinction on the edges $\beta_2\delta_i$ that are crossed by the path $p_{j+1}[h_{j+1}, t]$. (c) The region bounded by $\delta_4, \beta_2, \delta_3$ and the green polyline depicts the region ϕ. (d) The red edge is a short cut for p_{j+1}.

is good by invariant (iii). Let $\delta_1\gamma$, with $\gamma \neq \beta_2$, be the edge incident to δ_1 that lies on the boundary of h_j. In order to prove that $p_j[h_j, t]$ is good and h_j is not problematic with respect to this path, we suffices to show the following properties with respect to $p_j[h_j, t]$:

(P1) α has label $\perp$,
(P2) β_2 has label R,
(P3) δ_1 has label L, and
(P4) γ has label $\perp$.

For property (P1) consider the vertex α. Since $f_d^\delta = h_{j+1}$ it follows immediately that $p_{j+1}[h_{j+1}, t]$ and $p_j[h_{j+1}, t]$ do not cross an edge incident to α. Thus, the label of α with respect to $p_j[h_j, t]$ is by construction $\perp$.

For property (P2) recall that we assumed that β_2 has label R with respect $p_{j+1}[h_{j+1}, t]$. Moreover, $p_{j+1}[h_{j+1}, t]$ is, due to our invariant, good. Hence, by construction of p_j, β_2 has label R with respect to $p_j[h_j, t]$.

Let ρ be the region bounded by $p_{j+1}[h_j, f_1^\delta]$ and the edge dual to $\beta_2\delta_1$ that contains t. The vertex δ_1 lies, by the definition of ρ, outside of ρ and has therefore label L with respect to $p_j[h_j, t]$. For the vertex γ we distinguish the two cases $\gamma \neq \alpha$ and $\gamma = \alpha$. In case that $\alpha = \gamma$, it follows by Property (P1) that γ has the correct label. In case that $\alpha \neq \gamma$, we have that γ lies outside of ρ and therefore we have that γ has label $\perp$ with respect to $p_j[h_j, t]$. This concludes the proof of Property (P3) and (P4).

To conclude the proof, note that no vertex in the induced labeling by the path $p_j[h_j, t]$ has label LR. Thus, $p_j[h_j, t]$ is indeed good. Moreover, we have that $l_{p_j[h_j,t]}(\alpha) \neq l_{p_j[h_j,t]}(\delta_1)$ and $l_{p_j[h_j,t]}(\beta_2) \neq l_{p_j[h_j,t]}(\gamma)$. Hence, h_j is not problematic with respect to $p_j[h_j, t]$.

Case 3.3, $\beta_2\delta_3$ is crossed: Recall that $p_{j+1}[h_j, t]$ does not cross the edge $\beta_2\delta_d$. Hence, we have that $d = 4$. We obtain p_j from p_{j+1} by replacing the path $p_{j+1}[h_j, f_3^\delta]$ by the edges dual to $\alpha\beta_2$ and $\delta_4\beta_2$; see thick red path in Figure 8.7c. Since $p_{j+1}[h_j, t]$ crosses the edges dual to $\alpha\beta_1$ and $\beta_2\delta_3$, and p_{j+1} is a shortest st-path, we have that $|p_{j+1}| = |p_j|$, i.e., p_j is a shortest st-path. By construction we have that $p_{j+1}[s, h_j] = p_j[s, h_j]$. Note that $p_j[f_3^\delta, t] = p_{j+1}[f_3^\delta, t]$ is good by invariant (iii). Hence to finish this case we will prove the following properties:

(Q1) α has label R,
(Q2) β_2 has label L,
(Q3) δ_1 has label $\perp$,
(Q4) β_1 has label $\perp$,

Since δ_3 and δ_4 are on the boundary of h_{j+1}, there is a path q from δ_3 to δ_4 that is on the boundary of h_{j+1}. Let ϕ be the region that does not contain α and that is bounded by q and the edges $\delta_4\beta_2, \beta_2\delta_3$. Since $f_4^\delta = h_{j+1}$ and p is a shortest st-path, it follows that ϕ entirely contains $p_j[f_3^\delta, t]$ in its interior. Thus, Property (Q1) follows immediately from the construction of p_j. Since ϕ does not contain an edge incident to β_2 in its interior, Property (Q2) follows by construction of p_j. Moreover, the region does neither contain δ_1 nor β_1. Thus, the label of both vertices with respect to $p_j[h_j, t]$ is $\perp$. Therefore, Properties (Q3) and (Q4) hold.

Since the labeling induced by $p_j[h_j, t]$ does not contain LR, $p_j[h_j, t]$ is good. Moreover, we have that $l_{p_j[h_j,t]}(\beta_1) \neq l_{p_j[h_j,t]}(\beta_2)$ and, regardless of whether the edge $\beta_2\delta_1$ lies on the boundary of h_{j+1}, we have that $l_{p_j[h_j,t]}(\alpha) \neq l_{p_j[h_j,t]}(\delta_1)$. Therefore, h_j is not problematic with respect to p_j. ◁

This finishes the proof of the theorem. □

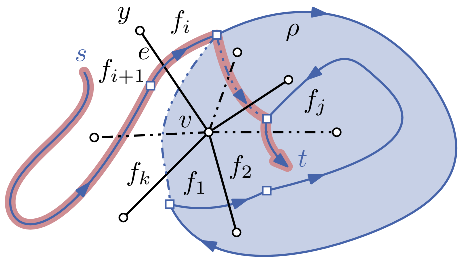


Figure 8.8: Inconsistent path around a degree k vertex. The shaded blue region depicts the region ρ.

Theorem 8.7. *Let $G = (V, E)$ be a planar embedded graph with maximum vertex-degree Δ and let p be a shortest st-path in $G^\star_{st}$ with $s, t \in V$. Then there is a consistent path of length at most $(\Delta - 2)|p|$.*

Proof. Let p be an st-path in $G^\star_{st}$. Assume that p is not consistent. Then there is a shortest prefix $p_2 = p[s, f_2] = p[s, f_1] \cdot f_1 f_2$ of p that is not consistent; refer to Figure 8.8. Let v be a vertex incident to the primal edge of $f_1 f_2$ with $l_{p_2}(v) = LR$. Without loss of generality let $f_1, f_2, \dots, f_k$ be the faces around v in counterclockwise order, i.e., v lies left of $f_1 f_2$.

Since p_2 is not consistent, there exists a second edge of p_2 that crosses a primal edge incident to v. Let e be the last edge of $p[s, f_1]$ that crosses a primal edge incident to v. Since p_2 is the shortest inconsistent prefix of p, v lies right of e, i.e., $e = f_{i+1} f_i$ for some i with $2 < i \leq k - 1$. Moreover, let f_j be the first vertex in clockwise order from f_i that lies on the path $p[f_2, t]$. Note that such a vertex f_j exists, since at the latest f_2 satisfies the condition.

Let q be the path $f_i f_{i-1} \cdots f_j$. We obtain a path p' from p by replacing $p[f_i, f_j]$ by q, i.e., $p' = p[s, f_i] \cdot q \cdot p[f_j, t]$. Note that, since f_j is the first vertex in clockwise order on $p[f_2, t]$, p' is a simple path. Since q does not contain the edges $f_k f_1$ and $f_1 f_2$, and $p[f_i, f_j]$ contains at least one edge, the path p' has length at most $|p| + (k - 2) - 1$. We claim that the prefix $p'_j = p'[s, f_j]$ is consistent.

Then, since $p'[f_j, t]$ is a subpath of $p[f_2, t]$ and $p'[s, f_j]$ is consistent, it follows that we have decreased the maximum length of a suffix of the path whose removal results in an inconsistent path. Since this suffix has initially length at most $|p|$, we inductively find a consistent st-path of length at most $(\Delta - 2)|p|$.

It remains to prove that $p'[s, f_j]$ is consistent. Since $p[s, f_2]$ is the shortest inconsistent prefix of p, the prefix $p[s, f_1]$ is consistent. Therefore, v is right of $p[s, f_i] = p'[s, f_i]$. By construction, v is right of q. Thus, we have $l_{p'_j}(v) = R$. The only vertices w of G with $l_{p'_j}(w) = LR$ can be neighbors of v, as otherwise $p[s, f_1]$ would not be consistent.

Consider the region ρ enclosed by the path $p[f_i, f_1]$ and $f_1, f_k, f_{k-1}, \dots, f_i$ that contains v; refer to Figure 8.8. The prefix $p[s, f_1] = p'[s, f_1]$ lies outside of ρ and the path q lies entirely in the interior of ρ. Moreover, in case that vw is crossed by $p'[s, f_i]$, w lies outside of ρ. On the other hand, if q crosses an edge vw, then w lies inside ρ. Thus, in both cases we immediately get that $l_{p'_j}(w) = L$. Therefore, the prefix $p'[s, f_j]$ is consistent. $\square$

8.4 Stretchable Shortest st-paths

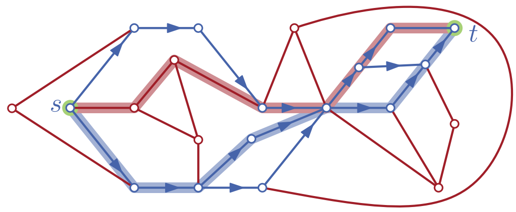
In Section 8.3 we showed that every shortest st-path in the extended dual $G^\star_{st}$ of a graph G with vertex degree at most 3 is consistent. For every graph of maximum degree 5, there is a shortest st-path $G^\star_{st}$ that is consistent. On the other hand, Figure 8.3 shows that, starting from degree 6, there are graphs whose shortest st-paths are not consistent. In this section we investigate the problem of deciding whether the extended dual $G^\star_{st}$ of a planar graph $G = (V, E)$ with a fixed combinatorial embedding and a fixed outer face contains a shortest st-path that is consistent. As a consequence of Proposition 8.1 and Theorem 8.3 this problem is in $\mathcal{NP}$.

Theorem 8.3 shows that finding a consistent st-path p in $G^\star_{st}$ is closely related to finding two edge-disjoint paths in G. Especially, we are interested in two edge-disjoint paths where the length of one is minimized. Eilam-Tzoreff [Eil98] proved that this problem is in general $\mathcal{NP}$-complete. In planar graphs the sum of the length of two vertex-disjoint paths can be minimized efficiently [KS10]. In general directed graphs the problem is $\mathcal{NP}$-hard [FHW80]. Finding two edge-disjoint paths in acyclic directed graphs is $\mathcal{NP}$-complete [EIS75].

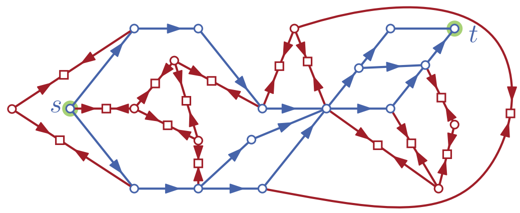
The closest relative to our problem is certainly the work of Eilam-Tzoreff. In fact their result can be modified to show that it is $\mathcal{NP}$-hard to decide whether a graph contains two edge-disjoint st-paths such that one of them is a shortest path. We study this problem in the planar setting, i.e., G is a planar embedded graph with outer face o, with the additional restriction that s and t lie on a common face o_{sp} of the subgraph G_{sp} of $G^\star_{st}$ that contains all shortest paths from s to t in $G^\star_{st}$.

Thus, we now consider the problem of finding a stretchable shortest st-path as an edge-disjoint path problem in $G^\star_{st}$. Our proof strategy consists of three steps. Step 1) We first show that the problem is equivalent to finding two edge-disjoint paths p and q in a directed graph $\overrightarrow{G}_{st}$ such that p is directed and q is undirected. Step 2) We modify $\overrightarrow{G}_{st}$ such that p is a path in a specific subgraph G_{sp} and q lies in the subgraph $\overline{G_{\mathrm{sp}}}$. These two graphs may share an edge set $\hat{E}$ such that each edge in $\hat{E}$ can be an edge of p or of q. Moreover, we find pairs of edges e and e' in $\hat{E}$ such that the path p in G_{sp} (the path q in $\overline{G_{\mathrm{sp}}}$) contains either e or e'. Step 3) Finally, we use these properties to reduce our problem to 2-SAT.

We begin with Step 1. Let $\overrightarrow{G}_{st} = (V' \cup \{s, t\}, E')$ be directed graph. A path $p = \langle v_1, v_2, \dots, v_k \rangle$ in $\overrightarrow{G}_{st}$ is a *directed path* if $v_i v_{i+1} \in E'$ for each $1 \leq i < k$. It is



(a)



(b)

Figure 8.9: (a) The graph $G^\star_{st}$ is the union of the blue shortest-path graph G_{sp} (without the directions) and the red exterior graph $G^\star_{st}$. (b) The modified graph $\overrightarrow{G}_{st}$ after Step 1.

undirected if either $v_iv_{i+1} \in E'$ or $v_{i+1}v_i \in E'$, for each $1 \le i < k$. The directed graph $\overrightarrow{G}_{st}$ is *st-friendly* if $G^\star_{st}$ contains a stretchable shortest st-path if and only if $\overrightarrow{G}_{st}$ contains a directed st-path p and an undirected p-friendly path p'. We obtain an st-friendly graph $\overrightarrow{G}_{st} = (\overrightarrow{V}, \overrightarrow{E})$ from $G^\star_{st}$ as follows. Denote by G_{sp} the directed acyclic graph that contains all shortest paths from s to t in $G^\star_{st} = (V, E)$. If an edge $uv \in E$ is an edge of G_{sp}, we add it to $\overrightarrow{G}_{st}$. For all remaining edges uv, we add a subdivision vertex x to $\overrightarrow{G}_{st}$ and add the directed edges xu, xv to $\overrightarrow{G}_{st}$ in this direction. We claim that $\overrightarrow{G}_{st}$ is st-friendly.

Proposition 8.8. *The graph $\overrightarrow{G}_{st}$ is st-friendly.*

Proof. Let p be a shortest st-path in $G^\star_{st}$ that is stretchable with respect to o. By Theorem 8.3 there is a p-friendly st-path p' in $G^\star_{st}$, i.e., p and p' are edge-disjoint and non-crossing, and p' contains $o^\star$. By construction, p corresponds to a directed path in G_{sp} and p' corresponds to an undirected path in $\overrightarrow{G}_{st}$.

Conversely, due to the directions of the edges xv, xu, every directed st-path q in $\overrightarrow{G}_{st}$ is a directed path in G_{sp}, and therefore it is a shortest st-path in $G^\star_{st}$, i.e., $p := q$. If there is an undirected p-friendly path q', we obtain a path p' from q' by contracting edges incident to split vertices x. Hence, $\overrightarrow{G}_{st}$ is st-friendly. □

We consider the following special case, where s and t lie on the outer face o_{sp} of the subgraph G_{sp} of $\overrightarrow{G}_{st}$. We denote by p_μ and p_λ the upper and lower st-path of G_{sp} on

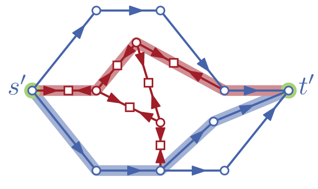


Figure 8.10: (a) The undirected path p' (red) uses vertices in the interior of G_{sp}. (b) Since the paths p (blue) and p' are not crossing, p' can be rerouted such that it does not use interior vertices.

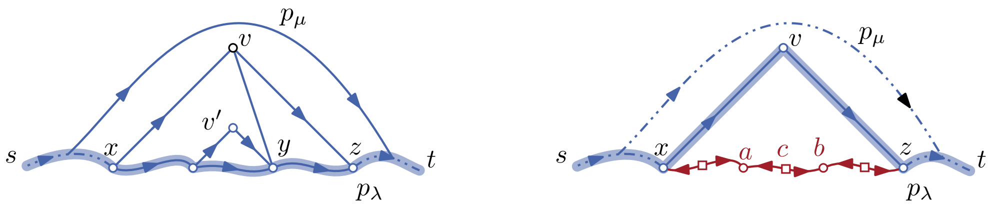


Figure 8.11: (a) The red directed path can be circumvented with the blue directed path via vertex v. (b) The red path consists of avoidable edges.

the boundary of o_{sp}. A vertex v of $\overrightarrow{G}_{st}$ is an *interior vertex* if v it lies in the exterior of o_{sp}. An edge uv of G_{sp} is an *interior edge* if u and v are interior vertices. An edge e of G_{sp} is a *chord* if both its endpoints lie on o_{sp} but e is not an edge on the boundary of o.

Lemma 8.9. *For a directed st-path p and an undirected p-friendly st-path p', there is an undirected p-friendly st-path p'' that does not use interior vertices of G_{sp}.*

Proof. If p' does not use interior vertices of G_{sp}, p' already satisfies the conditions for the path p'' and we set $p'' := p'$. Therefore, consider a p-friendly path p' that contains interior vertices of G_{sp}. By the definition of p-friendly paths, p and p' are non-crossing. Therefore there are two distinct vertices u, v on p_λ or on p_μ, say p_μ, such that the inner vertices of $p'[u, v]$ lie in the interior of G_{sp}; refer to Figure 8.10. Moreover, since p' and p are non-crossing, the region enclosed by $p'[u, v]$ and $p_\mu[u, v]$ does not contain a vertex of p in its interior. Therefore, we obtain p'' by iteratively replacing pieces in the form of $p'[u, v]$ by $p_\mu[u, v]$. Note that this operation preserves the property of p' that p' contains $o^\star$, since $o^\star$ can not be an interior vertex of G_{sp}, and is edge-disjoint from p. Hence, p'' is p-friendly. □

This finishes Step 1, and we continue with Step 2. In the following we iteratively simplify the structure of G_{sp} while preserving the st-friendliness of $\overrightarrow{G}_{st}$. Due to

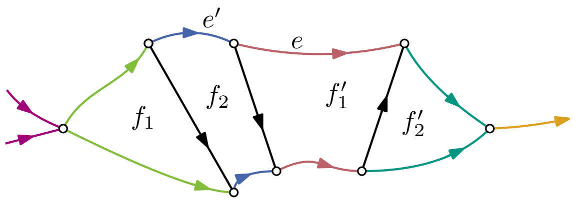


Figure 8.12: Interior partners decoded by color of a 2-edge connected component of the shortest path graph G_{sp}. Note that if p contains the edge e it also contains the edge e'.

Lemma 8.9, the graph G_{sp}/e, obtained from contracting an edge e of G_{sp}, is st-friendly, if e is an interior edge. This may generate a separating triangle xyz. Let v be a vertex in the interior of xyz and let p be a directed st-path that contains v. Then, p contains at least two vertices of x, y, z. Hence, p can be rerouted using an edge of xyz. Thus, the graph after removing all vertices in the interior of xyz is st-friendly. After contracting all interior edges of G_{sp}, each neighbor of an interior vertex of G_{sp} lies either on p_λ or on p_μ. The remaining edges are edges on $p_\lambda \cup p_\mu$ and chords.

Consider three vertices x, y, z that lie in this order on p_λ (p_μ) and two interior vertices v and v', with $xv, v'y, vz \in \overrightarrow{E}$; refer to Figure 8.11a. Note that v and v' can coincide. Then, every directed st-path p that contains y also contains x and z. Hence, p can be rerouted through the edges xv, vz and as a consequence of Lemma 8.9, the graph $G_{\text{sp}} - v'y$ is st-friendly. Analogously, if G_{sp} contains the edge yv', $G_{\text{sp}} - yv'$ remains st-friendly. We call such edges *circumventable*.

We refer to edges of a subpath $p_\lambda[x, z]$ ($p_\mu[x, z]$) as *avoidable* if there exists an interior vertex v with $xv, vz \in \overrightarrow{E}$ (Figure 8.11b). If there exists a directed path p that uses an avoidable edge ab it can be rerouted by replacing the corresponding path $p_\lambda[x, z]$ with the edges xv, vz. Thus, we can split the edge ab with a vertex c and we direct the resulting edges from c towards a and b, respectively, and remove the edge ab from $\overrightarrow{G}_{st}$. Finally, we iteratively contract edges incident to vertices with in- and out-degree 1, and we iteratively remove vertices of degree at most 1, except for s and t.

Since all interior edges of G_{sp} are contracted, circumventable interior edges are removed and avoidable edges are replaced, each 2-edge connected component of G_{sp} is an outerplanar graph whose weak dual (excluding the outer face) is a path; compare Figure 8.12. Each face f of G_{sp}, with $f \neq o_{\text{sp}}$, contains at least one edge e_λ of p_λ and one edge e_μ on p_μ. Moreover, every directed st-path contains either e_λ or e_μ. We refer to the edge sets $E_{f,\lambda} = E(f) \cap E(p_\lambda)$ and $E_{f,\mu} = E(f) \cap E(p_\mu)$ as *interior partners*.

Property 8.10. *Choosing a directed st-path in G_{sp} is equivalent to choosing for each face f of G_{sp} one of the interior partners $E_{f,\mu}$ or $E_{f,\lambda}$ such that the following condition holds. Let f_1, f_2 be two adjacent faces that are separated by a chord e that ends at p_λ (p_μ)*

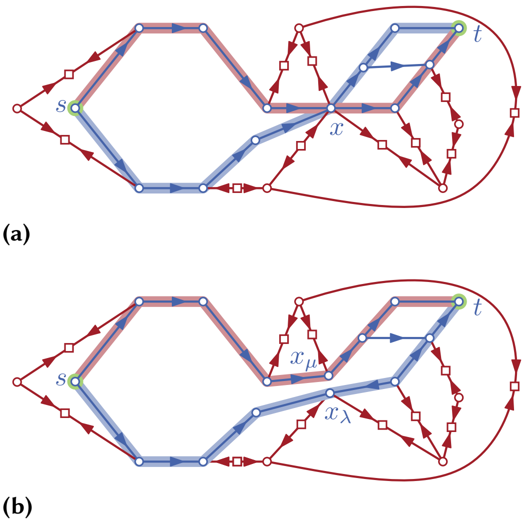


Figure 8.13: Splitting the vertex x into two copies x_λ and x_μ ensures that the red and blue paths do not cross.

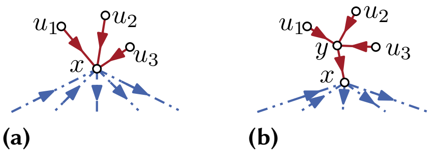
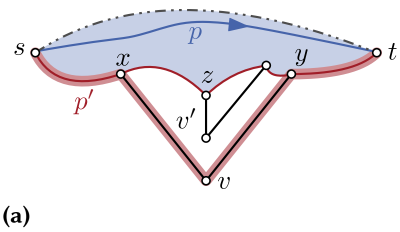


Figure 8.14: Splitting of a vertex x on the outer face o that is incident to an exterior edge $u_i x$.

such that f_1 is right of e (left of e), then the choice of $E_{f_2,\mu}$ ($E_{f_2,\lambda}$) implies the choice of $E_{f_1,\mu}$ ($E_{f_1,\lambda}$).

In the following, we modify the *exterior of* $\overrightarrow{G}_{st}$, i.e., $\overline{G_{\text{sp}}} = \overrightarrow{G}_{st} - E(G_{\text{sp}})$ where degree-0 vertices are removed, with the aim to obtain an analog property for the choice of the undirected path. We refer to edges of $\overline{G_{\text{sp}}}$ as *exterior edges*. A vertex in $V(\overline{G_{\text{sp}}}) \setminus V(G_{\text{sp}})$ is an *exterior vertex*.

Since the undirected path is not allowed to cross the directed path, we split each cut vertex x of G_{sp} into an upper copy x_μ and a lower copy x_λ; see Figure 8.13. We reconnect edges of p_λ and p_μ incident to x to x_λ and x_μ, respectively. Exterior edges incident to x that are embedded to the right of p_λ are reconnected to x_λ. Likewise, edges embedded to the left of p_μ are reconnected to x_μ. Note that this operation duplicates bridges of G_{sp}. Thus, we forbid the undirected path to traverse these duplicates. Observe that after this operation the outer face o_{sp} of G_{sp} is bounded by a simple cycle.



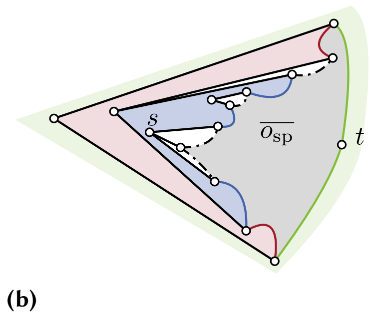
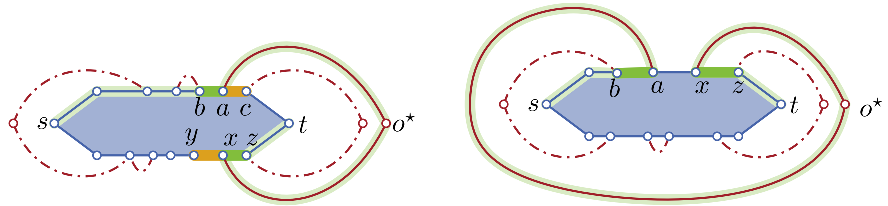


Figure 8.15: (a) If the undirected path p' (red) contains z, it can be rerouted along the path highlighted in light red so that it use vertex v. (b) The color coding of the faces indicate the exterior partners.

Let x be a vertex on o_{sp} that is incident to an exterior edge $u_i x$. In this case, we insert a vertex y to $\overrightarrow{G}_{st}$ and we remove each exterior edge $u_i x$ from $\overrightarrow{G}_{st}$ and insert as a replacement edges yx and $u_i y$; see Figure 8.14. We refer to the edge yx as a *barrier*. Since the barrier yx is directed from y to x, the modification preserves the st-friendliness of $\overrightarrow{G}_{st}$. We now exhaustively contract exterior edges that are not barrier edges, and remove vertices in the interior of separating triangles. In case of a contraction of an edge $ao^\star$ that is incident to the vertex dual to the outer face o of G, we remove a and reconnect its edges to $o^\star$.

Recall that s and t lie on the outer face o_{sp} of the subgraph G_{sp} of $\overrightarrow{G}_{st}$. Let v be an exterior vertex such that its neighbor x comes before its neighbor y on p_i with $i = \lambda, \mu$; refer to Figure 8.15a. Let z be a vertex between x and y on p_i that is connected to a vertex v' such that the edge $v'z$ (zv') lies in the interior of the region bounded by yvx and $p_i[x, y]$. Consider a directed st-path p in G_{sp} and an undirected p-friendly st-path p' in $\overrightarrow{G}_{st}$ that contains v'. Due to Lemma 8.9 we can assume, that p' does not contain an interior vertex of G_{sp}. Thus, it contains x and y. We obtain a new path p'' by replacing the subpath $p'[x, y]$ by vx, vy. Since vx, vy are exterior edges, p'' and p are edge-disjoint and non-crossing. Thus, the graph $\overrightarrow{G}_{st} - v'z$ ($\overrightarrow{G}_{st} - zv'$) is st-friendly. After removing all such edges, for any two neighbors x and y of an exterior vertex v, the paths $o_{\mathrm{sp}}[x, y]$ and $o_{\mathrm{sp}}[y, x]$ each contains either s or t. Hence, the region bounded by yvx and $o_{\mathrm{sp}}[x, y]$ contains a second exterior vertex v' if and only if $o_{\mathrm{sp}}[x, y]$ contains either s or t.

Hence, the dual of $\overline{G_{\mathrm{sp}}} + E(o_{\mathrm{sp}})$, with the dual vertex of complement $\overline{o_{\mathrm{sp}}}$ of o_{sp} and multi-edges removed, is a caterpillar C; refer to Figure 8.15b. In case that s or t is incident to an exterior vertex v, we can assume that the undirected path p' contains the edge sv (vt). Thus, for simplicity, we now assume that neither s nor t is connected



(a) $a \in V(p_\mu)$ and $x \in V(p_\lambda)$ **(b)** $a, x \in V(p_\mu)$

Figure 8.16: Every path (thick green path) that contains the outer face contains either the edges ac and yx (orange), or the edges ba and xz (green).

to an exterior vertex. Let a and b be the vertices in C whose primal faces are incident to s and t, respectively. Then every undirected st-path in $\overline{G_{\mathrm{sp}}} + E(o_{\mathrm{sp}})$ from s to t traverses the primal faces of the simple path q from a to b in C. Let f be a primal face of a vertex on q. Since we inserted the barrier edges to $\overrightarrow{G}_{st}$, every face contains at least one edge e_λ of p_λ and one edge e_μ of p_μ. Therefore, every undirected st-path in $\overline{G_{\mathrm{sp}}} + E(o_{\mathrm{sp}})$ either contains e_λ or e_μ. We refer to the sets $E_{f,\lambda} = E(f) \cap E(p_\lambda)$ and $E_{f,\mu} = E(f) \cap E(p_\mu)$ as *exterior partners*.

Property 8.11. *Choosing an undirected st-path in $\overline{G_{\mathrm{sp}}} + E(o_{\mathrm{sp}})$ is equivalent to choosing for each face $f \neq \overline{o_{\mathrm{sp}}}$ of $\overline{G_{\mathrm{sp}}} + E(o_{\mathrm{sp}})$ one of the exterior partners $E_{f,\lambda}$ or $E_{f,\mu}$.*

This finishes Step 2, and we proceed to Step 3. The problem of finding a directed st-path p and an undirected st-path p' in $\overrightarrow{G}_{st}$ reduces to a 2-SAT instance as follows. For each exterior and interior partner we introduce variables x_f and x_g, respectively, where f and g correspond to the faces of the partners. If x_f is true, p' contains the edge of $E_{f,\lambda}$, otherwise it contains $E_{f,\mu}$. The conditions on the choice of p in Property 8.10 can be formulated as implications. Let $E_{f,\mu}$ an $E_{f,\lambda}$ be exterior partners and let $E_{g,\mu}$ and $E_{g,\lambda}$ be interior partners. In case that $E_{f,\lambda} \cap E_{g,\lambda} \neq \emptyset$, either p can contain edges of $E_{g,\lambda}$ or p' can contains edges of $E_{f,\lambda}$ but not both. Thus, x_f and x_g are not allowed to be true at the same time, i.e., $x_f = \overline{x_g}$.

Finally, we have to ensure that p' contains the vertex $o^\star$ that is dual to the outer face o. Let $ao^\star$ and $xo^\star$ be the two edges incident to $o^\star$. Without loss of generality, assume that a lies on p_μ. We distinguish two cases based on whether x is on p_λ or on p_μ; refer to Figure 8.16. First assume that x is on p_λ. Let ba, ac and yx, xz be the edges incident to a and x, respectively, that lie on p_μ or p_λ. Every p-friendly st-path p' that contains $o^\star$, contains ba and xz, or yx and ac. Thus, let $E_{f_{ba},\mu}$ and $E_{f_{ac},\mu}$ be the edge set that contains the edges ba and ac, respectively. Analogously, let $E_{g_{yx},\lambda}$ and $E_{g_{xz},\lambda}$ be the set that contains yx, xz, respectively. Hence, we have that $x_{f_{ba}} = x_{g_{xz}}$ or $x_{f_{ac}} = x_{g_{yx}}$.

Now consider the case that a and x both lie on p_μ. Then let ba and xz be the edges on p_μ such that b precedes a and z succeeds x. Every path that contains $o^\star$ contains ba and bz. Let $E_{f,\mu}$ and $E_{g,\mu}$ be the sets that contain ba an bz, respectively. Then, we have that $x_f = x_g = \texttt{true}$.

Hence, we have the following Theorem.

Theorem 8.12. *If s and t lie on the outer face of G_{sp}, it is decidable in polynomial time whether $\overrightarrow{G}_{st}$ has a directed st-path p and a p-friendly st-path p'.*

Together Corollary 8.4 and Theorem 8.12 prove the following corollary.

Corollary 8.13. *If s and t lie on the outer face of G_{sp}, it is decidable in polynomial time whether $G^\star_{st}$ contains a shortest st-path that is stretchable with respect to the outer face of G.*

8.5 Parametrized Complexity of Short Consistent st-Paths

In this section we show that edge insertion can be solved in FPT time with respect to the minimum number of crossings of a straight-line drawing of $G + st$ where G is drawn without crossings and has the specified embedding. Let l be an arbitrary labeling of G. Observe that l defines a directed subgraph of $G^\star_{st}$ by removing each edge whose dual edge has endpoints with the same label and by directing all other edges e such that the endpoint of its primal edge left of e has label L and its other endpoint has label R. We denote this graph by $G^\star_{st}(l)$; edges incident to s or t are outgoing from s and incoming to t, respectively. Obviously, a shortest st-path in $G^\star_{st}(l)$ is compatible with l, and thus a corresponding drawing exists. Clearly, given the labeling l a shortest st-path in $G^\star_{st}(l)$ can be computed in linear time by a BFS.

Now assume that the length of a shortest consistent path in $G^\star_{st}$ is k. We propose a randomized FPT algorithm with running time $O(4^k n)$ for finding a shortest consistent path in $G^\star_{st}$, based on the color-coding technique [Cyg+15].

The algorithm works as follows. First, we pick a random labeling of G by labeling each vertex independently with L or R with probability $1/2$. We then compute a shortest path in $G^\star_{st}(l)$. We repeat this process 4^k times and report the shortest path found in all iterations.

Clearly the running time is $O(4^k n)$. Moreover, each reported path is consistent, and therefore the algorithm outputs only consistent paths. It remains to show that the algorithm finds a path of length k with constant probability.

Consider a single iteration of the procedure. If the random labeling l is compatible with p, then the algorithm finds a path of length k. Therefore the probability that our algorithm finds a consistent path of length k is at least as high as the probability that p is compatible with the random labeling l. Let $V_L, V_R \subseteq V$ denote the vertices of V that are left and right of p, respectively. Clearly it is $|V_L|, |V_R| \leq k$. A random labeling l is consistent with p if it labels all vertices in V_L with L and all vertices in V_R with R. Since vertices are labeled independently with probability $1/2$, it follows that $\Pr[p$ is consistent with $l] = (1/2)^{|V_L|} \cdot (1/2)^{|V_R|} \geq (1/2)^{2k} = (1/4)^k$.

Therefore, the probability that no path of length k is found in 4^k iterations is at most $(1-(1/4)^k)^{4^k}$, which is monotonically increasing and tends to $1/e \approx 0.368$. Thus the algorithm succeeds with a probability of $1 - 1/e \approx 0.632$. The success probability can be increased arbitrarily to $1-\delta$, $\delta > 0$ by repeating the algorithm $\log(1/\delta)$ times. The probability that each iteration fails is then bounded from above by $(1/e)^{\log 1/\delta} = 1/e^{\log 1/\delta} = \delta$. E.g., to reach a success probability of 99%, it suffices to do $\log 100 \leq 5$ repetitions. The algorithm can be derandomized with standard techniques [Cyg+15].

Theorem 8.14. *There is a randomized algorithm $\mathcal{A}$ that computes a consistent path of length k if one exists with a success probability of $1-\delta$. The running time of $\mathcal{A}$ is $O(\log(\delta^{-1})4^k n)$.*

8.6 Conclusion

We have shown that the problem of finding a short consistent st-paths in $G^\star_{st}$ is tractable in special cases and fixed-parameter tractable in general. Whether $G^\star_{st}$ has a short st-path stretchable with respect to a given outer face is equivalent to the question of whether $G^\star_{st}$ has two edge-disjoint and non-crossing st-paths, where the length of one path is minimized and the other contains the vertex dual to the outer face. Surprisingly, this is related to yet another purely graph theoretic problem: does a directed graph G have two edge-disjoint paths where one is directed and the other is only undirected? By the result of Eilam-Tzoreff [Eil98] the former problem is in general $\mathcal{NP}$-hard. For planar graphs the computational complexity of these problems remains an intriguing open question.

In Section 8.3 we showed that for each planar graph of maximum vertex degree 5 and for each pair s, t, there is a consistent shortest st-path. It is open whether this statement remains true if one asks for shortest st-paths that are stretchable with respect to the outer face. In this chapter, we only considered planar graphs with a fixed combinatorial embedding. Allowing for arbitrary embeddings opens new perspectives on the problem and is interesting future work.

9 Drawing Planar Clustered Graphs on Disks

Let $G = (V, E)$ be a planar graph and let $\mathcal{V}$ be a partition of V. We refer to the graphs induced by the vertex sets in $\mathcal{V}$ as *clusters*. Let $\mathcal{D}_C$ be an arrangement of pairwise disjoint disks with a bijection between the disks and the clusters. Akitaya et al. [AFT18] give an algorithm to test whether $(G, \mathcal{V})$ can be embedded onto $\mathcal{D}_C$ with the additional constraint that edges are routed through a set of pipes between the disks. If such an embedding exists, we prove that every clustered graph and every disk arrangement without pipe-disk intersections has a planar straight-line drawing where every vertex is embedded in the disk corresponding to its cluster. This result can be seen as an extension of the result by Alam et al. [Ala+15] who solely consider biconnected clusters. Moreover, we prove that it is $\mathcal{NP}$-hard to decide whether a clustered graph has such a straight-line drawing, if we permit pipe-disk intersections, even if all disks have unit size. This answers an open question of Angelini et al. [Ang+14].

The research of this chapter was initiated in the Bachelor thesis of Nina Zimbel [Zim17]. This chapter is based on joint work with Tamara Mchedlidze, Ignaz Rutter and Nina Zimbel [Mch+19b, Mch+19c].

9.1 Introduction

We study whether a clustered planar graph C has a planar straight-line drawing on a prescribed set of disks where each edge is allowed to intersect the boundary of each disk at most once. More formally, a *(flat) clustering* of a graph $G = (V, E)$ is a partition $\mathcal{V} = \{V_1, \ldots, V_k\}$ of the vertex set V. We refer to the pair $C = (G, \mathcal{V})$ as a *clustered graph* and the graphs $G_i = (V_i, E_i)$ induced by V_i as *clusters*. The set of edges E_i of a cluster G_i are *intra-cluster edges* and the set of edges with endpoints in different clusters are *inter-cluster edges*. A *disk arrangement* $\mathcal{D}_C = \{D_1, \ldots, D_k\}$ *of* C is a set of disks in the plane together with a bijective mapping $\mu(V_i) = D_i$ between the clusters $\mathcal{V}$ and the disks $\mathcal{D}$.

A *pipe* p_{ij} *of two clusters* V_i, V_j is the *convex hull* of the disks D_i and D_j, i.e., the smallest convex set of points containing D_i and D_j; see Figure 9.1. Observe that the boundary of p_{ij} is composed of two line segments u_{ij}, b_{ij} and two circular arcs. We refer to a *topological* planar drawing of G, i.e., the drawing of each edge is a curve, as an *embedding of* G. A $\mathcal{D}_C$*-framed embedding of* G is an embedding of G where each vertex $v \in V_i$ lies in the interior of the disk D_i and each edge uv, with $u \in V_i$ and $v \in V_j$, lies entirely in the pipe of V_i and V_j.

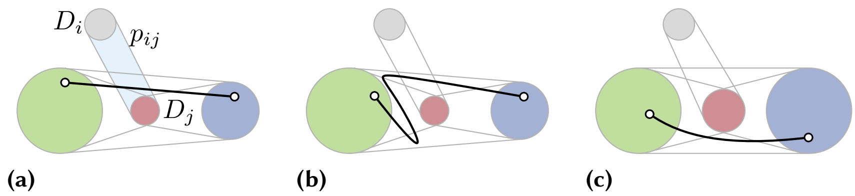


Figure 9.1: (a) The light-blue region shows the pipe p_{ij} of the disks D_i and D_j. An edge in a $\mathcal{D}_C$-framed straight-line drawing intersects the boundary of a pipe at most two times. Thus, the $\mathcal{D}_C$-framed embedding described in (b) does not correspond to $\mathcal{D}_C$-framed straight-line drawing. The drawing in (c) is not homeomorphic to (a), since the edge in (c) intersects different parts of the boundaries of the pipes.

Given a cluster planar graph C, a disk arrangement $\mathcal{D}_C$ of C and a $\mathcal{D}_C$-framed embedding ψ, Godau [God95] proves that it is $\mathcal{NP}$-hard to decide whether G has a $\mathcal{D}_C$-framed straight-line drawing Γ such that ψ is homeomorphic to Γ. The gadgets in the proof contain disks of size 0, i.e., the positions of some vertices are fixed. Moreover, there are disks that are entirely contained in a larger disk, i.e., there exist two disk $d_i, d_j, i \neq j$ with $d_i \subset d_j$. Angelini et al. [Ang+14] consider the case where G is not embedded but all disks have unit size. More formally, they show that given a planar graph G, it is $\mathcal{NP}$-hard to decide whether G has a $\mathcal{D}_C$-framed straight-line drawing. For unit disks, they leave the computational complexity of the question whether a $\mathcal{D}_C$-framed embedding has a corresponding $\mathcal{D}_C$-framed straight-line drawing as an open question. Banyassady et al. [Ban+17] show that this problem is $\mathcal{NP}$-hard in case that G is the *intersection graph* of $\mathcal{D}_C$, i.e., each vertex corresponds to a disk and two vertices are joined by an edge if the intersection of the corresponding disks is not empty.

The computational complexity of the following problem has not been considered: Given a cluster planar graph $C = (G, \mathcal{V})$, a set of pairwise disjoint disks $\mathcal{D}$ and a $\mathcal{D}_C$-framed embedding ψ, does C admit a $\mathcal{D}_C$-framed straight-line drawing of C that is *homeomorphic* to ψ. Thereby, we consider two $\mathcal{D}_C$-framed embeddings ψ, ψ' of C to be *homeomorphic* if (i) ψ and ψ' have the same combinatorial embedding and the same outer face, (ii) each edge e of G crosses a line segment u_{ij} (b_{ij}) of a pipe p_{ij} in ψ if and only if it crosses the respective line segment in ψ', (iii) and it does so in the same order. Observe that every edge in a $\mathcal{D}_C$-framed straight-line drawing intersects the boundary of a pipe at most twice; see Figure 9.1. Thus, in the following we assume as a necessary condition that an edge in a $\mathcal{D}_C$-framed embedding crosses the boundary of a pipe at most twice.

Related Work. Feng et al. [FCE95] introduced the notion of *clustered graphs* and *c-planarity*. A graph G together with a recursive partitioning of the vertex set is

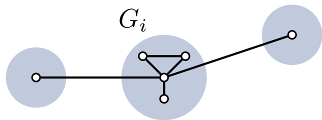


Figure 9.2: The cluster G_i cannot be augmented with edges such that G_i becomes biconnected.

considered to be a clustered graph. An embedding of G is *c-planar* if (i) each cluster c is drawn within a connected region R_c, (ii) two regions R_c, R_d intersect if and only if the cluster c contains the cluster d or vice versa, and (iii) every edge intersects the boundary of a region at most once. They prove that a c-planar embedding of a connected clustered graph can be computed in $O(n^2)$ time. It is an open question whether this result can be extended to disconnected clustered graphs. Many special cases of this problem have been considered [BR16].

Eades et al. [Ead+06] prove that every c-planar graph has a c-planar straight-line drawing where each cluster is drawn in a convex region. Angelini et al. [AFK11] strengthen this result by showing that every c-planar graph has a c-planar straight-line drawing in which every cluster is drawn in an axis-parallel rectangle. The result of Akitaya et al. [AFT18] implies that in $O(n \log n)$ time one can decide whether an abstract graph with a flat clustering has an embedding where each vertex lies in a prescribed topological disk and every edge is routed through a prescribed topological pipe. In general they ask whether a simplicial map φ of G onto a 2-manifold M is a *weak embedding*, i.e., for every $\epsilon > 0$, φ can be perturbed into an embedding ψ_ϵ with $||\varphi - \psi_\epsilon|| < \epsilon$.

Alam et al. [Ala+15] prove that it is $\mathcal{NP}$-hard to decide whether an embedded clustered graph has a c-planar straight-line drawing where every cluster is contained in a prescribed (thin) rectangle and edges have to pass through the interval common for both rectangles. Further, they prove that all instances with biconnected clusters always admit a solution. Their result implies that graphs of this class have $\mathcal{D}_C$-framed straight-line drawings.

Ribó [Rib06] shows that every embedded clustered graph where each cluster is a set of independent vertices has a straight-line drawing such that every cluster lies in a prescribed disk. In contrast to our setting Ribó allows an edge e to intersect a disk of a cluster G_i that does not contain an endpoint of e.

Contribution. We say that a disk arrangement $\mathcal{D}_C$ is *pipe-disk intersection free* if each pipe p_{ij} that contains an edge (i.e, $(V_i \times V_j) \cap E \neq \emptyset$) does not have an intersection with a disk d_k, where $k \neq i, j$. In Section 9.2, we prove that if the disk arrangement $\mathcal{D}_C$ is pipe-disk intersection free and each pair of disks is disjoint, then every clustered planar graph $(G, \mathcal{V})$ with a $\mathcal{D}_C$-framed embedding ψ has a $\mathcal{D}_C$-framed planar straight-line drawing homeomorphic to ψ. Taking the result of Akitaya et al. [AFT18] into account, our result can be used to test whether an abstract clustered graph with connected clusters has a $\mathcal{D}_C$-framed straight-line drawing. The example in Figure 9.2 shows that in general clusters cannot be augmented to be biconnected, if the embedding is fixed. Hence, our result is generalization of the result of Alam et al. [Ala+15]. In Section 9.3, we show that the problem is $\mathcal{NP}$-hard in the case that the disk arrangements is not pipe-disk intersection free. More specifically, we show that the problem is $\mathcal{NP}$-hard in case of arrangements of unit disks and as well as in the case of axis-aligned unit squares. This answers the aforementioned open question of Angelini et al. [Ang+14]. From now on we refer to a $\mathcal{D}_C$-framed straight-line drawing of G simply as a *$\mathcal{D}_C$-framed drawing* of G.

9.2 Drawing on Disk Arrangements that are Pipe-Disk Intersection Free

Let $C = (G, \mathcal{V})$ be a clustered planar graph, let $\mathcal{D}_C$ be a disk arrangement with pairwise disjoint disks that is pipe-disk intersection free, and let ψ be a $\mathcal{D}_C$-framed embedding of C. In this section we prove that C has a $\mathcal{D}_C$-framed drawing that is homeomorphic to ψ. We prove the statement by induction on the number of intra-cluster edges. In Lemma 9.1 we show that we can indeed reduce the number of intra-cluster edges by contracting intra-cluster edges. In Lemma 9.2, we prove that the statement is correct if the outer face of C is a triangle and C is *connected*, i.e., each cluster G_i is connected. In Theorem 9.3 we extend this result to clustered graphs whose clusters are not connected.

A triangle T in an embedded planar graph G is *separating* if the interior and exterior of T each contain a vertex of G. Let $e = uv$ be an intra-cluster edge of G that is not an edge of a separating triangle. We obtain a *contracted clustered graph C/e* of C by removing v from G and connecting the neighbors of v to u. We obtain a corresponding embedding ψ/e from ψ by routing the edges $vw \in E, w \neq u$ close to the original drawing of uv.

Lemma 9.1. *Let $C = (G, \mathcal{V})$ be a connected clustered planar graph, $\mathcal{D}_C$ be a disk arrangement with pairwise disjoint disks that is pipe-disk intersection free and let ψ be $\mathcal{D}_C$-framed embedding of C. Let e be an intra-cluster edge that is not an edge of a separating triangle. Then C has a $\mathcal{D}_C$-framed drawing that is homeomorphic to ψ if C/e has a $\mathcal{D}_C$-framed drawing that is homeomorphic to ψ/e.*

Proof. Let $e = uv$ and denote by $u_0, u_1, \ldots, u_k$ the neighbors of u and denote by $v_0, v_1, \ldots, v_l$ the neighbors of v in C in clockwise order; see Figure 9.3a. Without loss of generality, we assume that $u_0 = v$ and $v_0 = u$. Since e is not an edge of a separating triangle the set $I := \{u_2, \ldots, u_{k-1}\} \cap \{v_2, \ldots, v_{l-1}\}$ is empty. Denote by u the vertex

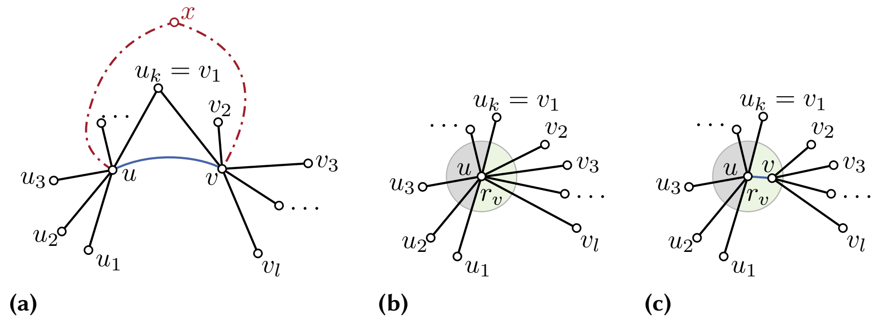


Figure 9.3: (a) Since uv is not an edge of a separating triangle edges xu, xv do not exist. (b) Moving u within disk d_u preserves the embedding of G/uv. (c) Drawing of G obtained from (b) by placing v in r_v.

obtained by the contraction of e. Let G_i be the cluster of u and v, and let D_i be the corresponding disk in $\mathcal{D}_C$.

Consider a $\mathcal{D}_C$-framed drawing Γ/e of C/e homeomorphic to ψ/e; see Figure 9.3b. Then there is a small disk $D_u \subset D_i$ around u such that for every point p in D_u moving u to p yields a $\mathcal{D}_C$-framed drawing that is homeomorphic to ψ/e.

We obtain a straight-line drawing Γ of C from Γ/e as follows; see Figure 9.3c. First, we remove the edges uv_i from Γ/e. The edges uu_1, uu_k partition D_u into two regions r_u, r_v such that the intersection of r_v with uu_i is empty for all $i \in \{2, \dots, k-1\}$. We place v in r_v and connect it to u and the vertices $v_1, \dots, v_l$. Since r_v is a subset of D_u and $I = \emptyset$, we have that the new drawing Γ is planar. Since v is placed in r_v, the edge uv is in between uu_1 and uu_k in the rotational order of edges around u. Hence, Γ is homeomorphic to ψ. Finally, Γ is a $\mathcal{D}_C$-framed drawing since, D_u is entirely contained in D_i and thus are u and v. □

Lemma 9.2. *Let C be a connected clustered graph with a triangular outer face T, let $\mathcal{D}_C$ be a disk arrangement with pairwise disjoint disks that is pipe-disk intersection free, and let ψ be a $\mathcal{D}_C$-framed embedding of C. Moreover, let Γ_T be a $\mathcal{D}_C$-framed drawing of T. Then C has a $\mathcal{D}_C$-framed drawing that is homeomorphic to ψ with the outer face drawn as Γ_T.*

Proof. We prove the theorem by induction on the number of intra-cluster edges.

First, consider the case that every intra-cluster edge of C is an edge on the boundary of the outer face. Note that there are at most three vertices in the interior of a single disk. Thus, C is either a triangle as depicted in Figure 9.4a and Figure 9.4b, or each cluster is a single vertex. Since $\mathcal{D}_C$ is pipe-disk intersection free, the graph in Figure 9.4a and Figure 9.4b C does not contain any further vertices. Let Γ be the drawing obtained from Γ_T by placing every vertex that does not lie on the outer face on the center point of its corresponding disk. Since $\mathcal{D}_C$ is a pipe-disk intersection free and Γ_T is

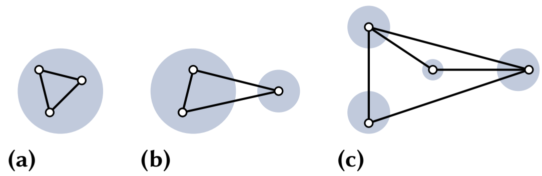


Figure 9.4: Instances with a triangular outer face that do not contain contractable intra-cluster edges.

convex, the resulting drawing is planar and thus a $\mathcal{D}_C$-framed drawing of C that is homeomorphic to the embedding ψ.

Let S be a separating triangle of C that splits C into two subgraphs C_{in} and C_{out} so that $C_{\text{in}} \cap C_{\text{out}} = S$ and the outer face of C_{out} and C coincide. Note that C_{in} and C_{out} are connected as otherwise C itself would not be connected. Then by the induction hypothesis C_{out} has the $\mathcal{D}_C$-framed drawing Γ_{out} with the outer face drawn as Γ_T and C_{in} has a $\mathcal{D}_C$-framed drawing Γ_{in} with the outer face drawn as $\Gamma_{\text{out}}[S]$, where $\Gamma_{\text{out}}[S]$ is the drawing of S in Γ_{out}. Then we obtain a $\mathcal{D}_C$-framed drawing of C by merging Γ_{in} and Γ_{out}.

Consider an intra-cluster edge e that does not lie on the boundary of the outer face and is not an edge of a separating triangle. Then by the induction hypothesis, C/e has a $\mathcal{D}_C$-framed drawing with the outer face drawn as Γ_T. It follows by Lemma 9.1 that C has a $\mathcal{D}_C$-framed drawing homeomorphic to ψ. □

Theorem 9.3. *Every clustered graph C with a $\mathcal{D}_C$-framed embedding ψ has a $\mathcal{D}_C$-framed drawing homeomorphic to ψ if the disk arrangement $\mathcal{D}_C$ is pairwise disjoint and pipe-disk intersection free.*

Proof. We obtain a clustered graph C' from C by adding a new triangle T to the graph and assigning each vertex of T to a newly constructed cluster. Let Γ_T be a drawing of T that contains all disks in $\mathcal{D}_C$ in its interior. We obtain a new disk arrangement $\mathcal{D}'_C$ from $\mathcal{D}_C$ by adding a sufficiently small disk for each vertex of Γ_T. The embedding ψ together with Γ_T is a $\mathcal{D}'_C$-framed embedding ψ' of C'.

According to Feng et al. [FCE95] there is a simple connected clustered graph C'' that contains C' as a subgraph whose embedding ψ'' is $\mathcal{D}_C$-framed and contains ψ'. By Lemma 9.2 there is a $\mathcal{D}_C$-framed drawing Γ'' of C'' homeomorphic to ψ'' with the outer face drawn as Γ_T. The drawing Γ'' contains a $\mathcal{D}_C$-framed drawing of C. □

9.3 Drawing on Arrangements with Pipe-Disk Intersections

In this section we study the following problem referred to as $\mathcal{D}_C$-FRAMED DRAWINGS WITH PIPE-DISK INTERSECTIONS. Given a planar clustered graph $C = (G, \mathcal{V})$, a disk arrangement $\mathcal{D}_C$ with pairwise disjoint disks that is not disk-pipe intersection free, and a $\mathcal{D}_C$-framed embedding ψ of C, is there a $\mathcal{D}_C$-framed drawing Γ that is homeomorphic to ψ?

Note that if the disks $\mathcal{D}_C$ are allowed to overlap and G is the intersection graph of $\mathcal{D}_C$, the problem is known to be $\mathcal{NP}$-hard [Ban+17]. Thus, in the following we require that the disks do not overlap, but there can be pipe-disk intersections. By Alam at al. [Ala+15] it follows that the problem restricted to thin touching rectangles instead of disks is $\mathcal{NP}$-hard. Their reduction heavily relies on the fact that the rectangles are thin. We strengthen this result and prove that in case that the rectangles are either axis-aligned unit squares or unit disks and are not allowed to touch the problem remains $\mathcal{NP}$-hard.

To prove $\mathcal{NP}$-hardness we reduce from PLANAR MONOTONE 3-SAT [BK12]. For each literal and clause we construct a clustered graph C with an arrangement of disks (squares) $\mathcal{D}_C$ of C such that each disk (square) contains exactly one vertex. We refer to these instances as *literal* and *clause gadgets*. In order to transport information from the literals to the clauses, we construct a *copy* and *inverter gadget*. For each gadget we first construct an arrangement of unit squares and state its important properties in this case. This is followed by the corresponding arrangement of unit disks. We emphasize the differences that have to be dealt with to preserve the properties of the gadgets when considering unit disks instead of unit squares. The design of the gadgets is inspired by Alam et al. [Ala+15], but the restriction to unit disks and squares rather than thin touching rectangles, requires a more complex construction and a careful placement of the geometric objects. The green and red regions in the figures of the gadget correspond to *positive* and *negative* drawings of the literal gadget. The green and red line segments indicate that for each truth assignment of the variables our gadgets indeed have $\mathcal{D}_C$-framed straight-line drawings. Negative versions of the literal and clause gadget are obtained by mirroring vertically. Hence, we assume that variables and clauses are positive. Each gadget covers a set of checkerboard cells. This simplifies the assembly of the gadgets in the final reduction. Note that in the following constructions all squares and disks will be of unit size. Moreover, we consider only axis-aligned squares.

9.3.1 Regulator

A line l separates the euclidean plane in two *half planes* h_a *and* h_b and we denote by $\overline{h_a}$ the complement of h_a. These half planes are *spanned by* l. We say that l *supports*

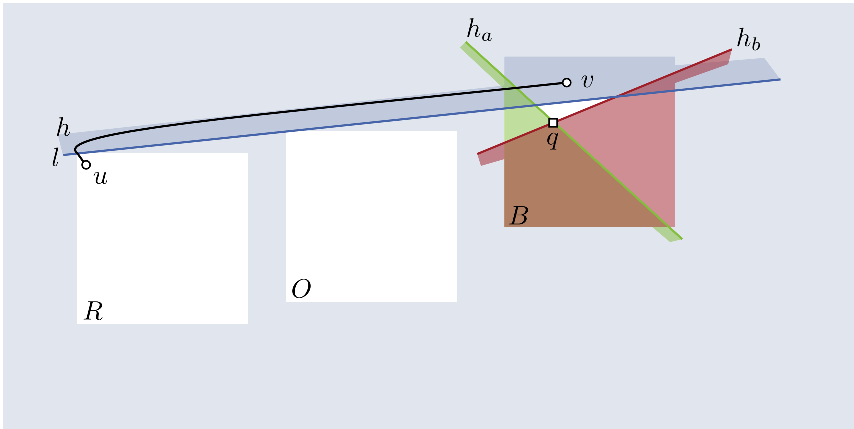


Figure 9.5: Regulator gadget

h_a (h_b). Let B be an axis-aligned square that contains a vertex v in its interior and let h_a, h_b be two half planes whose supporting lines have a unique intersection point q that lies in the interior of B; see Figure 9.5. We describe the construction of a gadget that restricts the feasible placements of v in a $\mathcal{D}_C$-framed drawing by a half plane h that excludes a placement of v in $h_a \cap h_b$ but allows for a placement in $h_a \cap B$ or $h_b \cap B$. Since q lies in the interior of B, there is a half plane h that does not contain q and for each $i = a, b$, $h \cap h_i \cap B$ is not empty, but $h \cap h_a \cap h_b = \emptyset$.

Let h, h_a, h_b and B as described before. We construct a *regulator gadget of v in B with respect to h_a and h_b* as follows. Let l_h be the supporting line of h. We create two axis-aligned squares R and O such that R, O and B intersect l_h in this order and h neither intersects the interior of R nor the interior of O. Place a vertex u in R and route an edge uv through $h \cup R \cup B$. In case that h instead of h_a and h_b is given, we refer to the gadget as the *regulator of v with respect to a (single) half plane h*.

Lemma 9.4. *Let W be a regulator gadget of v in B with respect to h_a and h_b. For every point $p_v \in h \cap B$ there is a $\mathcal{D}_C$-framed drawing Γ such that v lies on p_v. There is no $\mathcal{D}_C$-framed drawing of W such that v lies in $\overline{h} \cap B$.*

Proof. By construction of W, there is for every point $p_v \in h \cap B$ a $\mathcal{D}_C$-framed drawing Γ such that v lies on p_v.

The supporting line l_h of h intersects the boundary of R and does not intersect the interior of O. Let r and o be points in the intersection of l_h with R and O, respectively. Since Γ is homeomorphic to W the edge uv intersects l_h on the ray starting in o in the direction towards r. Therefore, u and v lie on different sides of l_h. Since $u \in R$, it follows that $v \in \overline{h}$. □

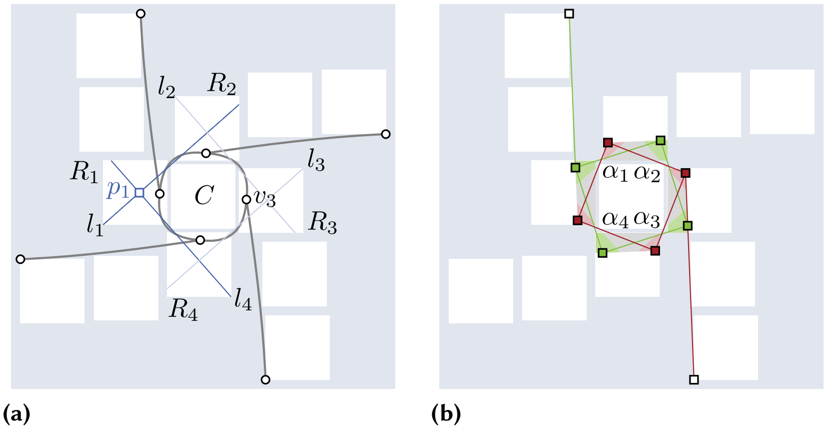


Figure 9.6: (a) The literal gadget. (b) The positive regions P_i are depicted in green and the negative regions N_i are red. The grey regions Q_i are infeasible. The green / red squared indicate that there are positive and negative realizations of the literal gadget.

We refer to the intersection $h \cap B$ as the *regulated region of v in B*. Thus, by the construction of W, the regulated region Q has a non-empty intersection with $h_a \cap B$ and $h_b \cap B$. Thus, by the lemma for each placement of v in $Q \cap h_i \cap B, i = a, b$, there is a $\mathcal{D}_C$-framed drawing. On the other hand, since $h \cap h_a \cap h_b \cap B = \emptyset$, there is no $\mathcal{D}_C$-framed drawing such that v lies in $h_a \cap h_b \cap B$.

9.3.2 Literal Gadget

In this section we construct a clustered graph C with an arrangement of squares $\mathcal{D}_C$ that models a literal u. The *positive literal gadget* is depicted in Figure 9.6a. We obtain the *negative literal gadget* by mirroring vertically.

The *center block* is a unit square C with corners $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ in clockwise order. For each corner α_i of C consider a line l_i that is tangent to C in α_i, i.e, $l_i \cap C = \{\alpha_i\}$. Let p_i be the intersection of the lines l_{i-1} and l_i where $l_0 = l_4$; refer to Figure 9.6a. Let $R_1, \ldots, R_4$ be four pairwise non-intersecting squares that are disjoint from C such that R_i contains p_i in its interior. We add a cycle $v_1v_2v_3v_4v_1$ to the graph such that $v_i \in R_i$. We refer to the vertex v_i as the *cycle vertex* of the *cycle block* R_i. For each i, let η_i be a half plane that contains R_{i+1} but does not intersect C. Within η_i we place a regulator W_i of v_i with respect to h_{i-1} and h_i, where h_i is the half plane spanned by l_i that does not contain C. This finishes the construction.

We now show that there exist two disjoint regions P_i and N_i in R_i that correspond to a positive and negative drawing of the literal gadget. Consider R_1 and its two adjacent squares R_4 and R_2. Let Q_i be the regulated region of R_i with respect to W_i. Then the intersection $I_1 := \overline{h_4} \cap \overline{h_1} \cap Q_1 \neq \emptyset$. We refer to I_1 as the *infeasible region*

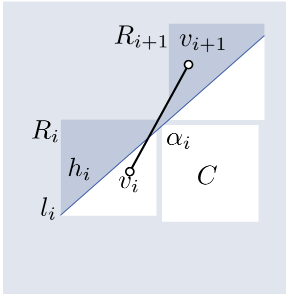


Figure 9.7: Since v_i does not lie in $h_i \cap R_i$ (green) and l_i is tangent to C, v_{i+1} lies in the $h_{i+1} \cap R_{i+1}$ (red).

of R_1. The intersection $h_1 \cap Q_1$ is the *positive region* P_1 *of* R_1. The region $h_4 \cap Q_1$ is the *negative region* N_1 *of* R_1. Regions P_1, N_1, I_1 are by construction not empty. The positive, negative and infeasible region of $R_i, i \neq 1$ are defined analogously.

Property 9.5. *If* Γ *is a* $\mathcal{D}_C$*-framed drawing of a literal gadget, then no cycle vertex* v_i *lies in the infeasible region of* R_i*. Moreover, either each cycle vertex* v_i *lies in the positive region* P_i *or each vertex* v_i *lies in the negative region* N_i*.*

Proof. Consider a $\mathcal{D}_C$-framed drawing Γ with an edge $v_i v_{i+1}$ such that v_i lies in $\overline{P_i}$, i.e., v_i lies in $\overline{h_i} \cap R_i$; see Figure9.7. We show that v_{i+1} lies in N_{i+1}. If v_{i+1} lies in $\overline{h_i}$, then v_i and v_{i+1} lie on the same side of l_i. Since l_i is tangent to α_i, $v_i v_{i+1}$ intersects C. It follows that v_{i+1} lies in h_i and therefore in the negative region N_{i+1}.

Assume that v_1 lies in its infeasible region I_1, then v_2 lies in N_2 by the above observation. Likewise, v_3, v_4, v_1 lie in N_3, N_4, N_1, respectively. This contradicts $N_1 \cap I_1 = \emptyset$. Similarly, we get that each vertex $v_i, i \neq 1$, cannot lie in the invisible region I_i. Thus, each v_i either lies in P_i or in N_i. Moreover, if one v_i lies in N_i the above observation yields that all of them lie in their negative region. □

The green and red squares in Figure 9.6a indicate that there is a positive and a negative realization of the literal gadget, i.e., there is a $\mathcal{D}_C$-framed drawing of the literal gadget where all cycle vertices lie either in a positive or in a negative region. In order to simplify the following constructions, we fix the position of the green and red squares as depicted. We refer to these positions as the *positive and negative placement* of the vertices v_i and denote them by $p^+_{X,i}$ and $p^-_{X,i}$. To reduce the notation, we drop the index i and simply refer to p^+_X and p^-_X as the positive and negative placements of the literal X. Thus, the literal gadget has the following property.

Property 9.6. *The positive and negative placements induce a* $\mathcal{D}_C$*-framed drawing of the literal gadget, respectively.*

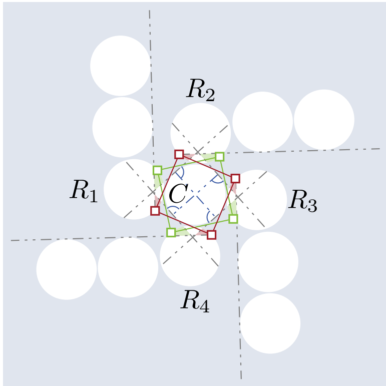


Figure 9.8: The literal gadget with unit disks. The endpoints of the blue segment in the interior of the central disk C are the points β_i.

Unit Disks

The construction of the literal gadget with unit disks follows the same principle as the construction using unit squares; see Figure 9.8. Only instead of the four corners α_i we choose four points β_i that are equally distributed along the boundary of the central disk. The position of the disk R_i have to be adjusted so that the it contains the intersection of the tangents of the central disks in the points β_{i-1} and β_i.

9.3.3 Copy and Inverter Gadget

In this section, we describe the copy and inverter gadget; see Figure 9.9. The copy gadget connects two positive or two negative literal gadgets X and Y such that a drawing of X is positive if and only if the drawing of Y is positive. Correspondingly, the inverter gadget connects a positive literal gadget X to a negative literal gadget Y such that the drawing of X is positive if and only if the drawing of Y is negative. The construction of the inverter and the copy gadget are symmetric.

Let X and Y be two positive literal gadgets whose center blocks are aligned on the x-axis with a sufficiently large distance. We construct the copy gadget that *connects* X and Y as follows. Let R_X and R_Y be the two cycle blocks of the literal gadgets X and Y, respectively, with minimal distance on the x-axis. For $A \in \{X, Y\}$, let P_A and N_A be the positive and negative regions of R_A. Since P_A and N_A are convex and their intersection is empty, there exists a half plane h_A that contains N_A but not P_A, and vice versa. In a reversed manner, we call h_A a *positive half-plane* h_Z^+ *of* A if it contains the negative region N_A, otherwise it is *negative* and we denote it by h_A^-.

Consider a positive half-plane h_X^+ of X and a negative half-plane h_Y^- of Y; refer to Figure 9.9a. We create two non-intersecting squares O_X^+ and O_Y^- that are contained in the intersection of $\overline{h_X^+}$ and $\overline{h_Y^-}$ such that a corner of O_X^+ and O_Y^- lie on the supporting line of h_X^+ and h_Y^-, respectively. Recall that we denote the complement of a half-plane h by $\overline{h}$. Let I be the intersection of the supporting lines of h_X^+ and h_Y^-. We place a square B with a vertex b in interior so that the intersection I lies in the interior of B. Additionally, we add a regulator of b with respect to h_X^+ and h_Y^- to exclude the intersection $h_X^+ \cap h_Y^-$ as feasible placement of b. We route the edges bv_X and bv_Y through $R_X \cup h_X^+ \cup B$ and $R_Y \cup h_Y^- \cup B$ respectively. This construction ensures that in a $\mathcal{D}_C$-framed drawing a placement of the vertex v_X in the positive region P_X excludes the possibility that the vertex v_Y lies in the negative region N_Y. In order to ensure that v_X cannot lie at the same time in N_X as v_Y in P_Y, we construct a square B' with respect to a negative half-plane h_X^- of X and a positive half-plane h_Y^+ of Y analogously to B. If the distance between X and Y is sufficiently large, we can ensure that the intersection of B and B' is empty. In the construction of the inverter gadget the square B is constructed with respect to h_X^+ and h_Y^+, and B' with respect to h_X^- and h_Y^-. We refer to the corresponding gadgets as *copy* and *inverter gadget*. We say that the copy and inverter gadget *connect* two literals.

Property 9.7. *Let Γ be a $\mathcal{D}_C$-framed drawing of two positive (negative) literals gadgets X and Y connected by a copy gadget. Then the $\mathcal{D}_C$-framed of X in Γ is positive if and only if the $\mathcal{D}_C$-framed drawing of Y is positive.*

Proof. By Property 9.5 the vertices v_X and v_Y of X and Y cannot lie in the infeasible regions of X and Y, respectively. Thus, similar to the proof of Lemma 9.5 we can assume for the sake of contradiction that the vertex b of the block B lies in the intersection of $\overline{h_X^+}$ and $\overline{h_Y^-}$. Thus, vertex v_X lies in the negative region of R_X and v_Y in the positive region of R_Y. But then vertex b' of the block B' lies in h_X^- and h_Y^+. However, this is not possible due to the regulator of b'. □

The same argumentation is applicable to the inverter gadget.

Property 9.8. *Let Γ be a $\mathcal{D}_C$-framed drawing of a positive literal gadget X and a negative literal gadget Y connected by an inverter gadget. Then the $\mathcal{D}_C$-framed drawing of X in Γ is positive if and only if the $\mathcal{D}_C$-framed drawing of Y is negative.*

The green and red squares in Figure 9.9b and in Figure 9.10 indicate that for a positive and a negative placement of X there is $\mathcal{D}_C$-framed drawing of copy and inverter gadget, respectively. Thus, the copy and inverter gadget have the following property.

Property 9.9. *The positive (negative) placement of two literals gadgets X, Y induces a $\mathcal{D}_C$-framed straight-line drawing of a copy [inverter] gadget that connects X and Y.*

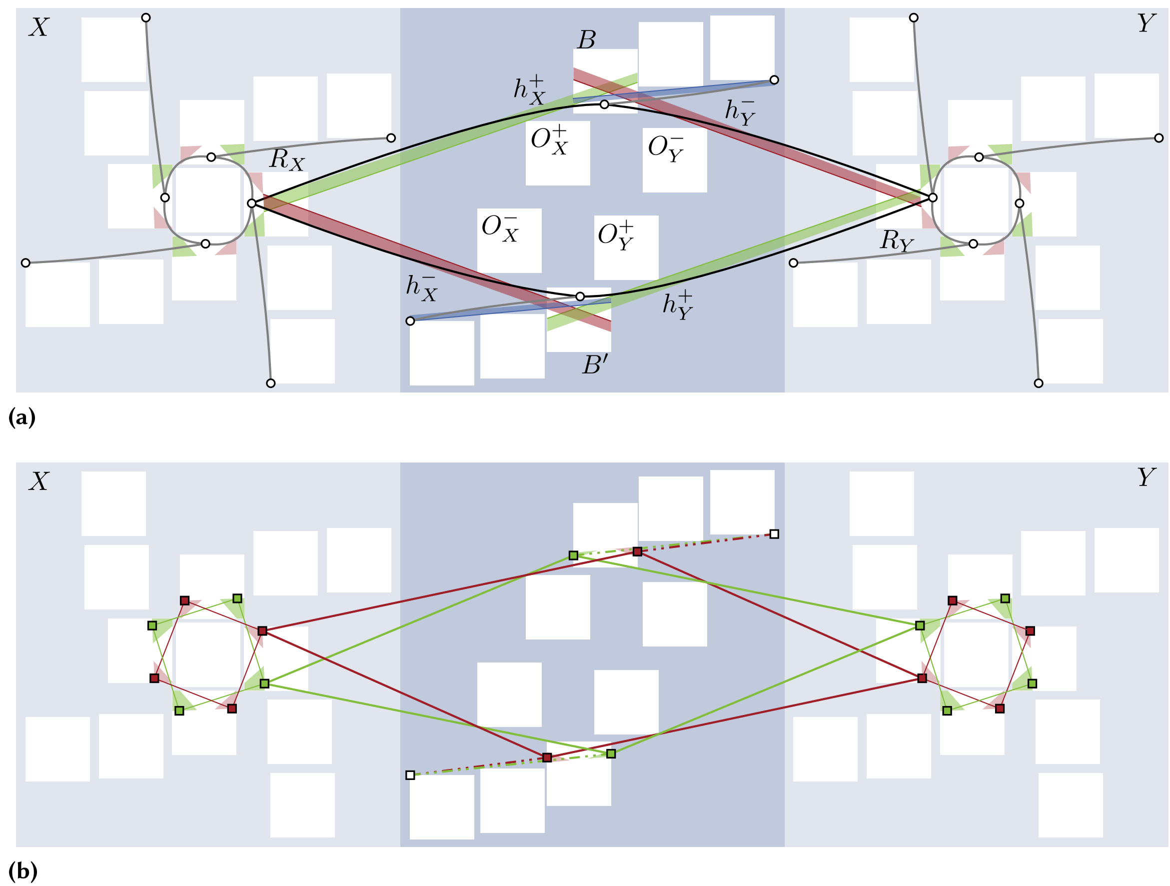


Figure 9.9: (a) The copy gadget. The thick transparent green and red lines depict the half planes h_X^+, h_Y^+ and h_X^-, h_Y^-, respectively. (b) Green and red regions depict positive and negative regions, respectively.

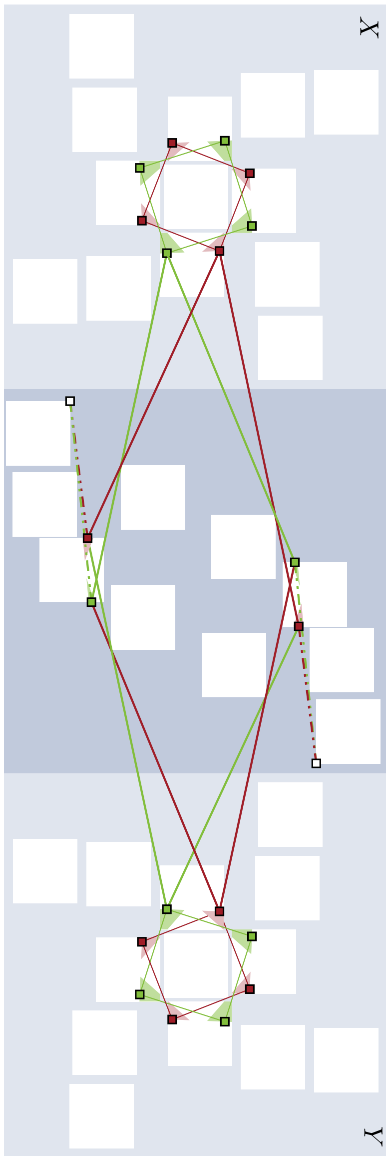


Figure 9.10: Positive and negative realizations of the inverter gadget.

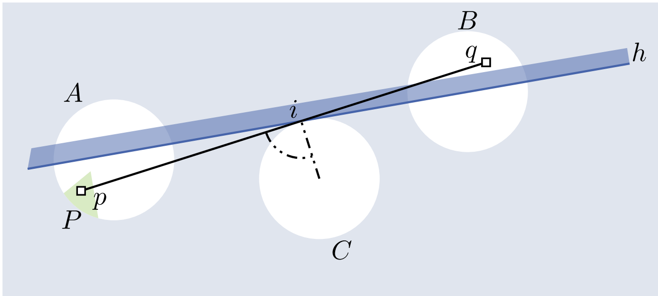


Figure 9.11: Observation

Unit Disks

Squares have the property that there is a set of tangents through a corner point of the square. On the other hand, at each point on the boundary of a disk the tangent to the disk is unique. The following observation helps to show that this restriction does not invalidate the correctness of the unit-disk gadgets.

Observation 9.10. *Let A and B be two disks and let P be a non-empty subset of A; see Figure 9.11. Moreover, let $p \in P$ and $q \in B$. Let i be the intersection of the segment pq and the supporting line of a half plane h that contains q and such that $h \cap P = \emptyset$. Let C be a disk such that pq is tangent to C in the point i. Let Q be the set of points in B so that for each $q' \in Q$ there is a point $p' \in P$ such that the segment $p'q'$ does not intersect C. Then Q is a strict subset of $h \cap B$.*

Recall that, for $A = X, Y$, let p_A^+ and p_A^- be the positive and negative placements of X and Y. Denote by h_A^+ and h_A^- the positive and negative half-planes, respectively, of the disk D_A; see Figure 9.12. Moreover, let q^+ and q^- be points in $h_X^+ \cap B$ and $h_Y^- \cap B$. Let O_X^+ (O_Y^-) be a disk such that $p_X^+ q^+$ ($p_Y^- q^-$) is tangent to O_X^+ (O_Y^-) in intersection of $p_X^+ q^+$ ($p_Y^- q^-$) with the supporting line of h_X^+ (h_Y^-). The disks O_X^- and O_Y^+ are positioned accordingly. The regulators of B and B' and Observation 9.10 ensure X has a positive $\mathcal{D}_C$-framed drawing if and only if Y has a positive $\mathcal{D}_C$-framed drawing.

9.3.4 Clause Gadget

We construct a *clause gadget* with respect to three positive literal gadgets X, Y, Z arranged as depicted in Figure 9.13. The negative clause gadget, i.e., a clause with three negative literal gadgets, is obtained by mirroring vertically.

We construct the clause gadget in two steps. First, we place a *transition block* T_A close to each literal gadget $A \in \{X, Y, Z\}$. In the second step, we connect the transition block to a vertex k in a *clause block* K such that for every placement of k in K at least one drawing of the literal gadgets has to be positive.

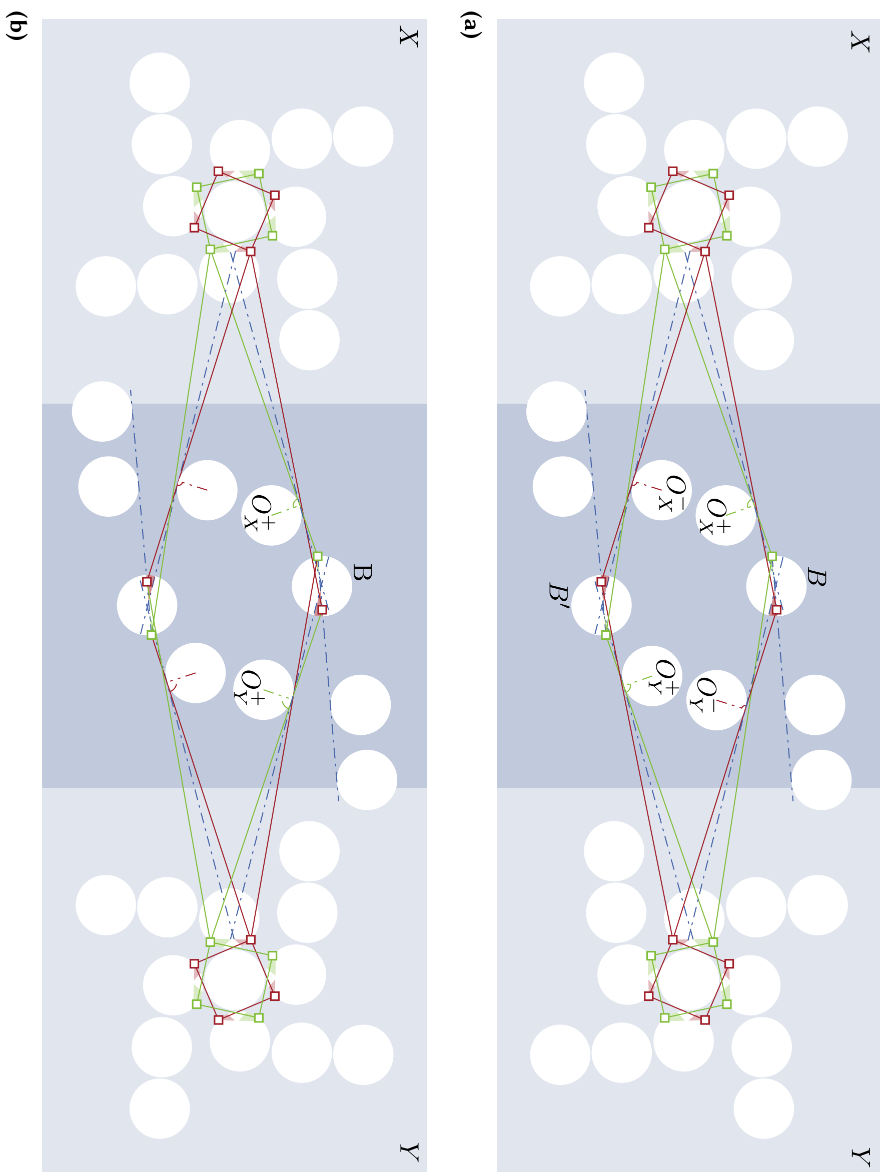


Figure 9.12: (a) Disk copy gadget. (b) Disk inverter gadget.

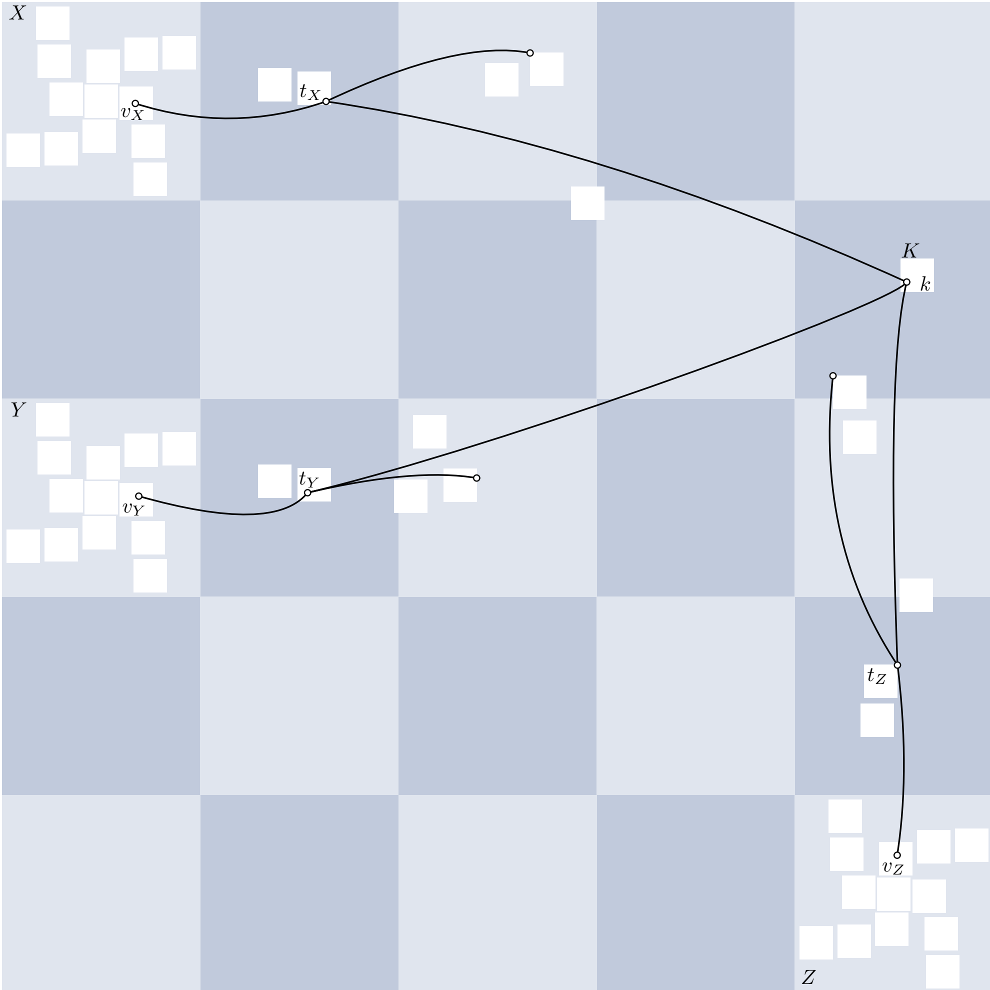


Figure 9.13: Clause gadget.

Consider the literal gadget X and let R_X be the rightmost cycle block of X. Let h_X^- be a negative half-plane of R_X, i.e., h_X^- contains the positive region but not the negative region; refer to Figure 9.14. We now place a transition block T_X such that the intersection $T_X \cap h_X^-$ has small area. Recall that p_X^+ and p_X^- denote the positive and negative placements of X, respectively. Let q_X^- be a point in $T_X \cap h_X^-$. Note that, in the following l^- and l^+ denote lines and *not* the half-planes left or right of a line l. Let i be the intersection point of the supporting line l_X^- of h_X^- and the line segment $p_X^- q_X^-$. We place a square Q_X such that l_X^- is tangent to Q_X at point i. We place a *transition vertex* t_X in the interior of T_X and route the edge $v_X t_X$ through $h_X^- \cup T_X \cup R_X$, where $v_X \in R_X$.

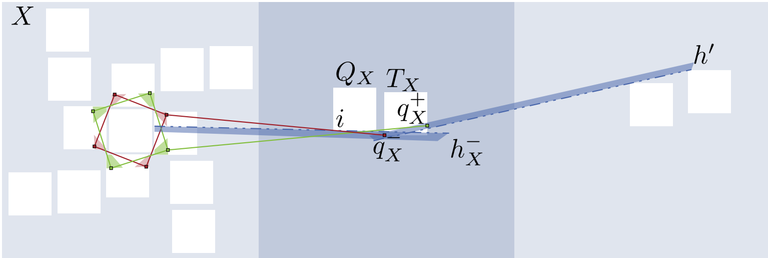
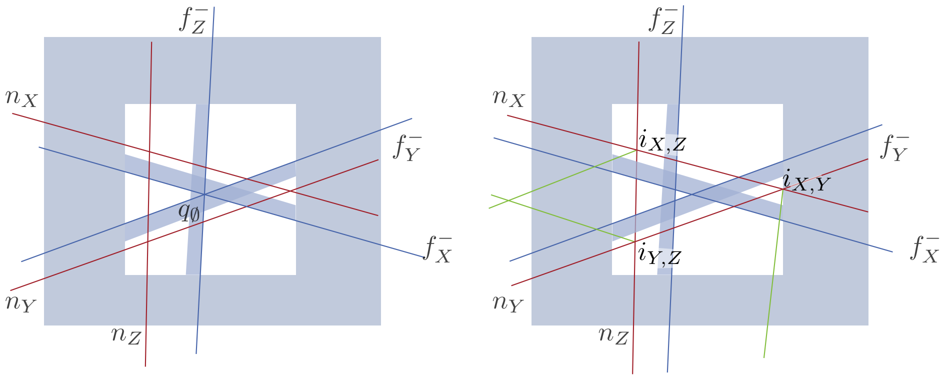


Figure 9.14: $\mathcal{D}_C$-framed drawings of the transition block of literal X



(a) **(b)**

Figure 9.15: (a) Initial placement of $q_\emptyset$ and the corresponding half planes h_A^-. (b) Setting after perturbing h_A^-. The green segments indicate that each $q_A^+, A = X, Y, Z$ can be connected with a line segment to each intersection $i_{X,Y}, i_{X,Z}, i_{Y,Z}$.

Observe that q_X^- allows for a negative drawing of X; see Figure 9.14. Let l_X^+ be a line that is tangent to Q_X and that contains p_X^+. Then each point on l_X^+ that lies in the interior of T_X allows for a positive drawing of X. Let q_X^+ be the point on l_X^+ that maximizes the distance to q_X^-. We refer to q_X^+ and q_X^- as the *positive* and *negative placements of* t_X, respectively. Further, if X has a negative drawing, then t_X lies in the region $h_X^- \cap T_X$. In order to reduce the visibility of t_X in case that X is negative, we place a regulator gadget of T_X with respect to a half plane h' as follows. Let h' be a half plane that contains q_X^- and q_X^+ and reduces the possible positions of t_X in this case to $h' \cap h_X^- \cap T_X$; see Figure 9.14. In the following, we refer to $h' \cap h_X^- \cap T_X$ as the *negative region* of T_X. The transition blocks of Y and Z are constructed analogously with only minor changes.

Let K be the clause block as depicted in Figure 9.13. Further, let $q_\emptyset$ be a point in the interior of K. Let f_A^-, for $A \in \{X, Y, Z\}$, be half planes such that the supporting lines

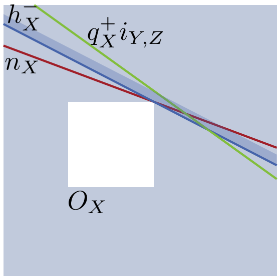


Figure 9.16: Intersection pattern near square O_X.

of all three half planes intersect at $q_\emptyset$ and such that f_A^- does not contain the negative region N_A of the transition block T_A; see Figure 9.15. Recall that q_A^- denotes the negative placement of t_A in the transition block T_A. Let n_X and n_Y be two lines whose intersection lies in the interior of $f_X^- \cap f_Y^- \cap K$ and that contain q_X^- and q_Y^-, respectively. Moreover, denote by n_Z a line that contains q_Z^- with a non-empty intersection with $f_Z^- \cap K$. We position a square O_A that is tangent to n_A at point $n_A \cap l_A^-$, where l_A^- is the supporting line of f_A^- and such that the intersection of the interior of O_A and f_A^- is empty. By construction of O_A all three literals gadgets X, Y, Z have negative $\mathcal{D}_C$-framed drawings if and only if k lies on $q_\emptyset$. Slightly perturbing the positions of the squares O_A ensures that the intersection $f_X^- \cap f_Y^- \cap f_Z^-$ is empty. Denote by $i_{B,C}$, for $B, C \in \{X, Y, Z\}$ with $B \neq C$, the intersection of n_B and n_C. To ensure that there are the necessary positive and negative drawings, the perturbation operation has to ensure that the intersection of the line through q_X^+ and $i_{X,Y}$ with n_B and f_X^- has the pattern as depicted in Figure 9.16 and correspondingly for the literals Y and Z. Thus, the clause gadget has the following property.

Property 9.11. *There is no $\mathcal{D}_C$-framed drawing of the clause gadget such that the $\mathcal{D}_C$-framed drawing of each literal gadget is negative. For all remaining combinations of positive and negative drawings of the literal gadgets X, Y and Z there is a $\mathcal{D}_C$-framed drawing of the clause gadget.*

Unit Disks

We utilize Observation 9.10 twice to ensure the correctness of the clause gadget with unit disks. First, recall that the square Q_X in Figure 9.14 is positioned such that Q_X is tangent to the supporting line of h_X^- and the line l^- that contains p_X^- and q_X^-, in point i. Replacing Q_X by a disk Q'_X that such that the disk is tangent to l^- in point i ensures that q_X^- corresponds to negative drawing of X. Moreover, by Observation 9.10 the set of points that possibly allow for a negative drawing is a subset of $h_X^- \cap Q'_X$. The disks Q'_Y, Q'_Z are constructed analogously.

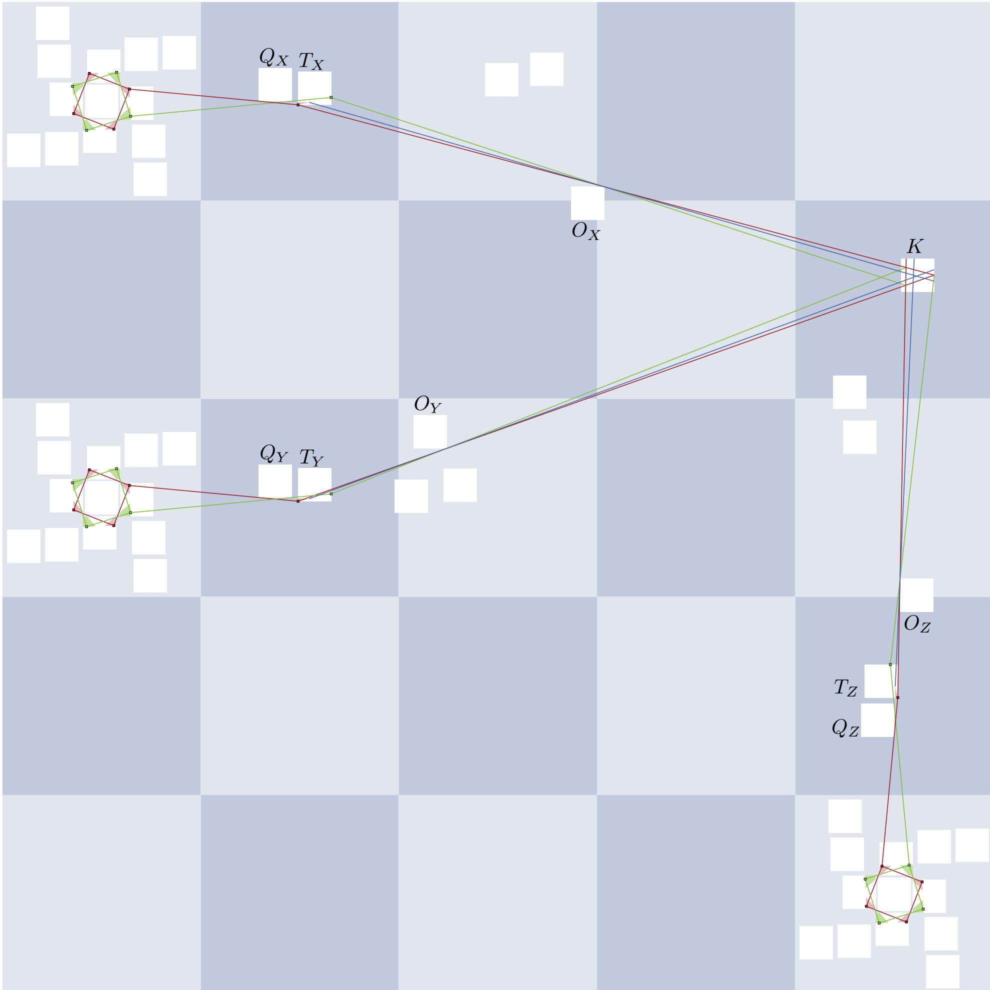


Figure 9.17: $\mathcal{D}_C$-framed drawings of the clause gadgets.

Second, recall the construction of the square O_A for $A = X, Y, Z$. The disk O'_A that corresponds to the square O_A is placed such that the line n_A is tangent to O'_A in the intersection of n_A with the supporting line of the half place f_A^-. Figure 9.18 shows the final clause gadget with unit disks.

9.3.5 Reduction

A 3-SAT instance (U, C) on a set U of n boolean variables and m clauses C is *monotone* if each clause either contains only positive or only negative literals. It is *planar* if the bipartite graph $G_{U,C} = (U \cup C, \{uc \mid u \in c \text{ or } \overline{u} \in c \text{ with } u \in U \text{ and } c \in C\})$ is

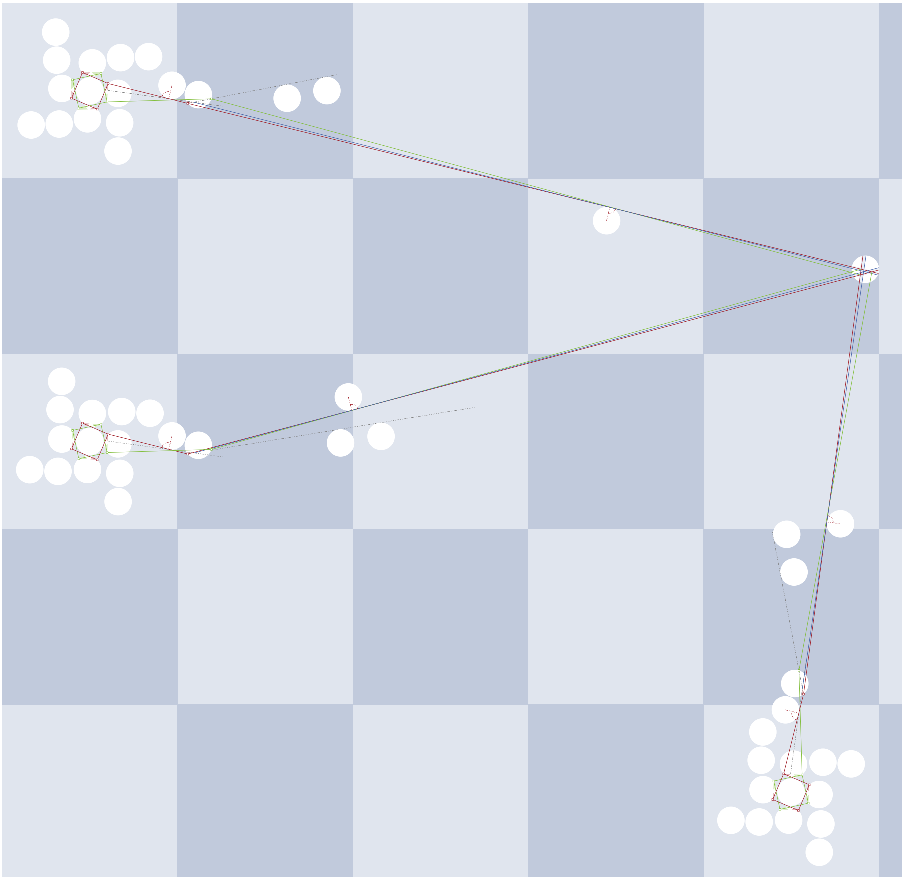


Figure 9.18: Clause Construction

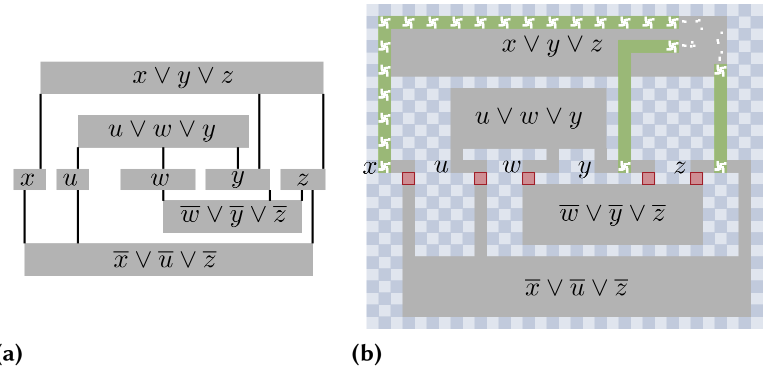


Figure 9.19: Example of planar monotone 3-SAT instance with a corresponding rectilinear representation.

planar. A *rectilinear representation* of a monotone planar 3-SAT instance is a drawing of $G_{U,C}$ where each vertex is represented as an axis-aligned rectangle and the edges are vertical line segments touching their endpoints; see Figure 9.19a. Further, all vertices corresponding to variables lie on a common line l, the positive and negative clauses are separated by l. The problem Monotone Planar 3-SAT asks whether a monotone planar 3-SAT instance with a given rectilinear representation is satisfiable. De Berg and Khosravi [BK12] proved that Monotone Planar 3-SAT is $\mathcal{NP}$-complete. We use this problem to show that the $\mathcal{D}_C$-framed Drawings with Pipe-Disk Intersections problem is $\mathcal{NP}$-hard.

In the following a disk d_k is an *obstacle of a pipe* p_{ij}, for i, j with $i, j \neq k$, if $d_k \cap p_{ij} \neq \emptyset$. The *obstacle number of a pipe* p_{ij} is the number of obstacles of p_{ij}. The *obstacle number of a disk arrangement* $\mathcal{D}_C$ is the maximum obstacle number over all pipes p_{ij} with $V_i \times V_j \cap E \neq \emptyset$.

Theorem 9.12. *The problem $\mathcal{D}_C$-framed Drawings with Pipe-Disk Intersections with axis-aligned unit squares and unit disks is $\mathcal{NP}$-hard even when the clustered graph C has maximum vertex degree 5 and its obstacle number is* 2*.*

Proof. Let (U, C) be a planar monotone 3-SAT instance with a rectilinear representation Π. Let l be a horizontal or vertical line that intersects Π. The line l splits Π into two drawings Π_L and Π_R that are left and right of l, respectively. For a positive factor x, we obtain from Π a new rectilinear representation by moving Π_R x units to the right. We fill the resulting gap between Π_L and Π_R with infinitely many copies of $l \cap \Pi$. This operation of *stretching the drawing at line l* allows us to do the following necessary modifications.

In the following we modify Π to fit on a checkerboard of $O(|C|)$ rows and columns where each column has width d and every row has height d. A row or column is *odd* if its index is an odd number, otherwise it is *even*. The pair (i, j) refers to the cell in column i and row j. We align all vertices corresponding to variables in the rectilinear representation in row 0 so that the leftmost variable vertex is in column 1; refer to Figure 9.19b. The width of each rectangle r_u of variable u is increased to cover $2 \cdot n_u - 1$ columns, where n_u is the number of occurrences of u and $\bar{u}$ in C. To ensure that each r_u starts in an odd column, we increase the distance between two consecutive variables so that the number of columns between the variables is odd and is at least three. Since we are able to add an arbitrary number of columns between two consecutive variables, we can assume without loss of generality that no two edges of the rectilinear representation share a column and that their columns are odd. We adapt the rectangle of a clause so that it covers five rows and at least six columns, and so that its left and right sides are aligned with the leftmost and rightmost incoming edges, respectively. Note that the positive clauses lie in rows with positive indices and the negative clauses in rows with negative indices. Each operation adds at most a constant number of columns and rows per vertex and per edge to the layout. Thus, the width and height of the final layout is in $O(|C|)$. Further, it can be computed in time polynomial in $|C|$.

In the following we construct a planar embedded graph C and an arrangement of squares $\mathcal{D}_C$ of C. We use the modified rectilinear layout to locally replace the variable by a sequence of positive and negative literals connected by either a copy or an inverter gadget. Clauses are replaced with the clause gadget and then connected with a sequence of literals and copy gadget to the respective literal in the variable.

Observe that the literal gadget is constructed so that all its squares fit in a larger square S. The copy and inverter gadget together with two literals is constructed so that they fit in rectangle three times the size of S. The clause gadget fits in a rectangle of width six times the size of the square S and its height is five times the height of S.

We assume that the size of the square S and the size of the squares of the checkerboard coincide. Let $r = 0$ be the row that contains the variable vertices. Every column contains at most one edge of the rectilinear representation. Thus, we place a positive literal gadget in cell (i, r) if the edge in column i connects a variable u to a positive clause. Otherwise, we place a negative literal gadget in cell (i, r). Since every edge of the rectilinear representation lies in an odd column, we can connect two literals of the same variable by either a copy or inverter gadget depending on whether both literals are positive or negative, or one is positive and the other negative.

We substitute an edge e of the rectilinear representation that connects a variable to a positive clause as follows. Let i be the column of e. If the cell (i, r_e) is covered by e and r_e is odd, we place a positive literal gadget in cell (i, r_e). The copy gadget can be rotated in order to connect a literal gadget in cell (i, r_e) to a literal gadget in a cell $(i, r_e + 2)$.

Let R_c be the rectangle that corresponds to the positive clause c in the modified rectilinear representation. We insert a clause gadget in R_c and justify it on the right of it so that the literal gadget Z lies in an odd column. Note that by the construction of clause gadget this fixes the position of the corresponding literal gadgets X and Y. Finally, the literal gadget X, Y and Z can be connected to their variables x, y and z as depicted in Figure 9.19b. A negative clause is obtained by vertically mirroring the construction of a positive clause.

We now argue that the embedding of the graph C is planar and that the pairwise intersections of squares in the arrangement $\mathcal{D}_C$ are empty. Observe that, every gadget is entirely embedded in the modified rectilinear representation. Recall that the rectilinear representation is planar and all gadget are placed in disjoint cells. Therefore, the pairwise intersection of squares in $\mathcal{D}_C$ is empty. Moreover, each literal gadget is planar embedded in a single cell, each clause is embedded in a rectangle that covers five rows and six columns, and finally each copy and inverter gadget together with its two literal gadget is embedded in either a single row and 3 columns or in 3 rows and a single column. Thus, since the modified rectilinear representation is planar and the pairwise intersections of squares in $\mathcal{D}_C$ are empty, the graph C has a planar embedding. Finally, the maximal vertex degree of the literal gadget is three, the maximal degree a clause gadget is four. Connecting two literal gadgets by copy or inverter gadget increases the maximum vertex degree of C to five. Further, the obstacle number of the clause gadget is one and the obstacle number of the literal, copy and the inverter gadget is two.

It is left to show that the layout can be computed in polynomial time. As already argued the modified rectilinear representation Π of the monotone planar 3-SAT instance can be computed polynomial time. Moreover, the height and width of Π is linear in $|C|$. Thus, we inserted a number of gadgets linear in $|C|$. Further, the coordinates of each gadget are independent of the instance (U, C), thus overall the representation of the final arrangement $\mathcal{D}_C$ is polynomial in $|U|$ and $|C|$. Placing a single gadget requires polynomial time, thus overall the clustered graph C and the arrangement $\mathcal{D}_C$ of squares can be computed in polynomial time.

Correctness. Assume that (U, C) is satisfiable. Depending on whether a variable u is true or false, we place all cycle vertices on a positive placement of a positive literal gadget and on the negative placement of negative literal gadget of the variable. Correspondingly, if u is false, we place the vertices on the negative and positive placements, respectively. By Property 9.6, the placement induces a $\mathcal{D}_C$-framed drawing of all literal gadgets. Property 9.9 ensures that the copy and the inverter gadgets have a $\mathcal{D}_C$-framed drawing. Since at least one variable of each clause is true, there is a $\mathcal{D}_C$-framed drawing of each clause gadget by Property 9.11.

Now consider the clustered graph C has a $\mathcal{D}_C$-framed drawing. Let X and Y be two positive literal gadgets or two negative literal gadgets connected with a copy gadget. By Property 9.7, a drawing of X is positive if and only if the drawing of Y is positive. Property 9.8 ensures that the drawing of a positive literal gadget X is positive if and only if the drawing of the negative literal gadget Y is negative, in case that both are joined with an inverter gadget. Further, Property 9.5 states that each cycle vertex lies either in a positive or negative region. Thus, the truth value of a variable u can be consistently determined by any drawing of a positive or negative literal gadget of u. By Property 9.11, the clause gadget has no $\mathcal{D}_C$-framed drawing of the clause gadget such that all literal gadgets have a negative drawing. Thus, the truth assignment indeed satisfies C. □

9.4 Conclusion

We proved that every clustered planar graph with a pipe-disk intersection free disk arrangement $\mathcal{D}_C$ and with a $\mathcal{D}_C$-framed embedding ψ has a $\mathcal{D}_C$-framed straight-line drawing homeomorphic to ψ. In case of arrangements of unit disks and unit squares with pipe-disk intersections the problem becomes $\mathcal{NP}$-hard. This answers an open question of Angelini et al. [Ang+14]. We are not aware whether the problem is known to be in $\mathcal{NP}$. Due to the geometric nature of the problem, we ask whether techniques developed by Abrahamsen et al. [AAM18] can be used to prove $\exists\mathbb{R}$-hardness. The cycles in the literal and copy gadget are crucial for our reduction. Thus, we ask whether the problem becomes tractable for restricted graph classes, e.g., trees, outerplanar graphs, or planar graphs that have maximum vertex degree 4.

10 Aligned Drawings of Planar Graphs

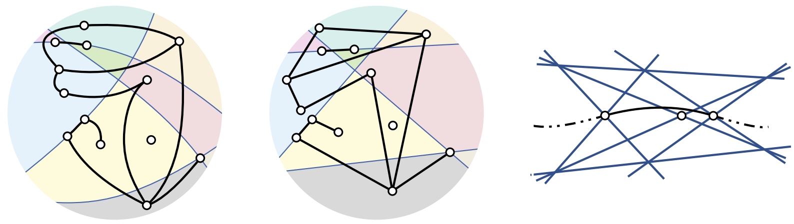
Let G be a graph that is topologically embedded in the plane and let $\mathcal{A}$ be an arrangement of pseudolines intersecting the drawing of G. An *aligned* drawing of G and $\mathcal{A}$ is a planar polyline drawing Γ of G with an arrangement A of lines so that Γ and A are homeomorphic to G and $\mathcal{A}$. We show that if $\mathcal{A}$ is stretchable and every edge e either entirely lies on a pseudoline or it has at most one intersection with $\mathcal{A}$, then G and $\mathcal{A}$ have a straight-line aligned drawing. In order to prove this result, we strengthen a result of Da Lozzo et al. [Da +18], and prove that a planar graph G and a single pseudoline $\mathcal{C}$ have an aligned drawing with a prescribed convex drawing of the outer face. We also study the less restrictive version of the alignment problem with respect to one line, where only a set of vertices is given and we need to determine whether they can be collinear. We show that the problem is $\mathcal{NP}$-complete but fixed-parameter tractable.

This chapter is based on joint work with Tamara Mchedlidze, Ignaz Rutter and Peter Stumpf [Mch+19a, MRR18a, MRR18b].

10.1 Introduction

Two fundamental primitives for highlighting structural properties of a graph in a drawing are *alignment* of vertices such that they are collinear, and geometric *separation* of unrelated graph parts, e.g., by a straight line. Both these techniques have been previously considered from a theoretical point of view in the case of planar straight-line drawings.

Da Lozzo et al. [Da +18] study the problem of producing a planar straight-line drawing of a given embedded graph $G = (V, E)$ (i.e., G has a fixed combinatorial embedding and a fixed outer face) such that a given set $S \subseteq V$ of vertices is collinear. It is clear that if such a drawing exists, then the line containing the vertices in S is a simple curve starting and ending at infinity that for each edge e of G either fully contains e or intersects e in at most one point, which may be an endpoint. We call such a curve a *pseudoline with respect to* G. Da Lozzo et al. [Da +18] show that this is a full characterization of the alignment problem, i.e., a planar straight-line drawing where the vertices in S are collinear exists if and only if there exists a pseudoline $\mathcal{L}$ with respect to G that contains the vertices in S. However, the computational complexity of deciding whether such a pseudoline exists is an open problem, which we consider in this chapter.



(a) (b) (c)

Figure 10.1: (Pseudo-) Lines are depicted as blue curves, edges are black. The color of the cells indicates the bijection ϕ between the cells of $\mathcal{A}$ and A. Aligned drawing (b) of a 2-aligned planar embedded graph (a). (c) A non-stretchable arrangement of 9 pseudolines (blue and black), which can be seen as a stretchable arrangement of 8 pseudolines (blue) and an edge (black solid).

Likewise, for the problem of separation, Biedl et al. [BKM98] considered so-called HH-drawings where, given an embedded graph $G = (V, E)$ and a partition $V = A \cup B$, one seeks a y-monotone planar polyline drawing of G with few bends in which A and B can be separated by a line. Again, it turns out that such a drawing exists if there exists a pseudoline $\mathcal{L}$ with respect to G such that the vertices in A and B are separated by $\mathcal{L}$. As a side-result Cano et al. [CTU14] extend the result of Biedl et al. to planar straight-line drawings with a given star-shaped outer face.

The aforementioned results of Da Lozzo et al. [Da +18] show that given a pseudoline $\mathcal{L}$ with respect to G one can always find a planar straight-line drawing of G such that the vertices on $\mathcal{L}$ are collinear and the vertices contained in the half-planes defined by $\mathcal{L}$ are separated by a line L. In other words, a topological configuration consisting of a planar embedded graph G and a pseudoline with respect to G can always be stretched. In this chapter, we initiate the study of this stretchability problem with more than one given pseudoline.

More formally, a pair $(G, \mathcal{A})$ is a k-*aligned graph* if $G = (V, E)$ is a planar embedded graph and $\mathcal{A} = \{C_1, \dots, C_k\}$ is an arrangement of (pairwise intersecting) pseudolines with respect to G. In case that every pair of distinct pseudolines intersect at most once, we refer to $\mathcal{A}$ as a *pseudoline arrangement*. If the number k of pseudolines is clear from the context, we drop it from the notation and simply speak of *aligned graphs*. For 1-aligned graphs we write (G, C) instead of $(G, \{C\})$. Let $A = \{L_1, \dots, L_k\}$ be a line arrangement and Γ be a planar drawing of G. A tuple (Γ, A) is an *aligned drawing of* $(G, \mathcal{A})$ if and only if the arrangement of the union of Γ and A is homeomorphic to the arrangement of the union of G and $\mathcal{A}$. A (pseudo)-line arrangement divides the plane into a set of *cells* $C_1, C_2, \dots, C_\ell$. If A is homeomorphic to $\mathcal{A}$, then there is a bijection ϕ between the cells of $\mathcal{A}$ and the cells of A. If (Γ, A) is an aligned drawing of $(G, \mathcal{A})$, then it has the following properties; refer to Figure 10.1(a-b). (i) The arrangement of A is homeomorphic to the arrangement of $\mathcal{A}$ (i.e., $\mathcal{A}$ is *stretchable* to A), (ii) Γ is homeomorphic to the planar embedding of G, (iii) the intersection of each vertex v and each edge e with a cell C of $\mathcal{A}$ is non-empty if and only if the intersection of v and e with $\phi(C)$ in (Γ, A), respectively, is non-empty, (iv) if an edge uv (directed from u to v) intersects a sequence of cells $C_1, C_2, \ldots, C_r$ in this order, then uv intersects in (Γ, A) the cells $\phi(C_1), \phi(C_2), \ldots, \phi(C_r)$ in this order, and (v) each line L_i intersects in Γ the same vertices and edges as C_i in G, and it does so in the same order. We focus on straight-line aligned drawings. For brevity, unless stated otherwise, the term aligned drawing refers to a straight-line drawing throughout this chapter.

Note that the stretchability of $\mathcal{A}$ is a necessary condition for the existence of an aligned drawing. Since testing stretchability is $\mathcal{NP}$-hard [Mnë88, Sho91], we assume that a geometric realization A of $\mathcal{A}$ is provided. Line arrangements of size up to 8 are always stretchable [P80], and only starting from nine lines non-stretchable arrangements exist; see the Pappus configuration [Lev26] in Figure 10.1c. This figure also illustrates an example of an 8-aligned graph with a single edge that does not have an aligned drawing. It is conceivable that in practical applications, e.g., stemming from user interactions, the number of lines to stretch is small, justifying the stretchability assumption.

The aligned drawing convention generalizes the problems studied by Da Lozzo et al. and Biedl et al. who focused on the case of a single line. We study a natural extension of their setting and ask for alignment on general line arrangements.

In addition to the strongly related works mentioned above, there are several other works that are related to the alignment of vertices in drawings. Ravsky and Verbitsky [RV11] used the fact that 2-trees have a drawing with at least $n/30$ collinear vertices to show that at least $\sqrt{n/30}$ vertices of a 2-tree can be fixed to arbitrary positions. Dujmović [Duj17] shows that every n-vertex planar graph $G = (V, E)$ has a planar straight-line drawing such that $\Omega(\sqrt{n})$ vertices are aligned, and Da Lozzo et al. [Da+18] show that in planar treewidth-3 and planar treewidth-k graphs, one can align $\Theta(n)$ and $\Omega(k^2)$ vertices, respectively. Chaplik et al. [Cha+16] study the problem of drawing planar graphs such that all edges can be covered by k lines. They show that it is $\mathcal{NP}$-hard to decide whether such a drawing exists. The computational complexity of deciding whether there exists a drawing where all vertices lie on k lines is an open problem [Cha+17]. Drawings of graphs on n lines where a mapping between the vertices and the lines is provided have been studied by Dujmović et al. [DL13b, Duj+11].

Contribution & Outline. After introducing notation in Section 10.2, we first study the topological setting where we are given a planar graph G and a set S of vertices to align in Section 10.3. We show that it is $\mathcal{NP}$-complete to decide whether S is alignable. On the positive side, we prove that this problem is fixed-parameter tractable (FPT) with respect to $|S|$. Afterwards, in Section 10.4, we consider the geometric

Table 10.1: Families of aligned graphs that always have an aligned drawing are marked with ✓. The symbol ✗ indicates that for this particular class, there is an aligned graph that does not have an aligned drawing.

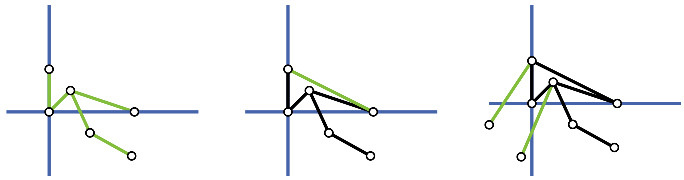
alignment complexity	k	drawable
$(0, \perp, \perp)$	≥ 1	✓ – Planarity
$(0, 0, 0)$	≥ 1	✓ – Theorem 10.1
$(1, 0, \perp)$	≥ 1	✓ – Theorem 10.19
$(1, 0, 0)$	2	✓ – Theorem 10.27
$(1, 1, 0)$	2	✗ – Theorem 10.2
$(1, 0, 0)$	k	open
$(\perp, \perp, 2)$ $(\perp, 3, \perp)$ $(4, \perp, \perp)$	≥ 8	✗ – Figure 10.1(c)

setting where we seek an aligned drawing of an aligned graph. Based on our proof strategy in Section 10.4.1, we strengthen the result of Da Lozzo et al. and Biedl et al. in Section 10.4.2, and show that there exists a 1-aligned drawing of G with a given convex drawing of the outer face. In Section 10.4.3 we consider k-aligned graphs with a stretchable pseudoline arrangement, where every edge e either entirely lies on a pseudoline or intersects at most one pseudoline, which can either be in the interior or an endpoint of e. We utilize the result of Section 10.4.2 to prove that every such k-aligned graph has an aligned drawing, for any value of k. Already in Section 10.2 we prove that not every 2-aligned graph has an aligned drawing. In Section 10.4.4, we show that special subclass of 2-aligned graphs always have an aligned drawing. In the preliminaries we define the *alignment complexity* of an aligned graph. It is a triple that indicates how many intersections an edge has with the pseudoline arrangement depending on the number of endpoints that lie on a pseudoline. Table 10.1 summarizes the results of our chapter.

10.2 Preliminaries

Let $\mathcal{A}$ be a pseudoline arrangement with k pseudolines $C_1, \ldots, C_k$ and $(G, \mathcal{A})$ be an aligned graph with n vertices. The set of cells in $\mathcal{A}$ is denoted by cells($\mathcal{A}$). A cell is *empty* if it does not contain a vertex of G. Removing from a pseudoline its intersections with other pseudolines gives its *pseudosegments*.

Let $G = (V, E)$ be a planar embedded graph with vertex set V and edge set E. We call $v \in V$ *interior* if v does not lie on the boundary of the outer face of G. An edge $e \in E$ is *interior* if e does not lie entirely on the boundary of the outer face of G. An interior edge is a *chord* if it connects two vertices on the outer face. A point p of an edge e is an



(a) $(1, 0, \perp)$ **(b)** $(1, 0, 0)$ **(c)** $(2, 1, 0)$

Figure 10.2: Examples for the alignment complexity of an aligned graph.

interior point of e if p is not an endpoint of e. A *triangulation* is a biconnected planar embedded graph whose inner faces are all triangles and whose outer face is bounded by a simple cycle. A *triangulation* of a graph G is a triangulation that contains G as a subgraph. A *k-aligned triangulation* of $(G, \mathcal{A})$ is a k-aligned graph $(G_T, \mathcal{A})$ with G_T being a triangulation of G. A graph G' is a *subdivision* of G if G' is obtained by placing *subdivision vertices* on edges of G. For an abstract graph G and an edge e of G the graph G/e is obtained from G by contracting e and merging the resulting multiple edges and removing self-loops. Routing the edges incident to e close to e yields a planar embedding of G/e in case of a planar embedded graph G. A *k-wheel* is a simple cycle C with k vertices on the outer face and one additional interior vertex that has an edge to each vertex in C. Let Γ be a drawing of G and let C be a cycle in G. We denote with $\Gamma[C]$ the drawing of C in Γ. Let T be a separating triangle in G and let V_{in} and V_{out} be the vertices in the interior and exterior of T, respectively. We refer to the graphs induced by $T \cup V_{\text{in}}$ and $T \cup V_{\text{out}}$ as the *split components of* T and denote them by G_{in} and G_{out}.

A vertex is *C_i-aligned* (or simply *aligned* to C_i) if it lies on the pseudoline C_i. A vertex that is not aligned is *free*. An edge e is *C_i-aligned* (or simply *aligned*) if it completely lies on C_i. Let E_{aligned} be the set of all aligned edges. An *intersection vertex* lies on the intersection of two pseudolines C_i and C_j. A non-aligned edge is *i-anchored* ($i = 0, 1, 2$) if i of its endpoints are aligned to distinct pseudolines. An C-aligned edge is *i-anchored* ($i = 0, 1, 2$) if i of its endpoints are aligned to distinct pseudolines which are different from C. For example, the single aligned edge in Figure 10.2a is 1-anchored. Let E_i be the set of i-anchored edges; note that, the set of edges is the disjoint union $E_0 \cup E_1 \cup E_2$. An edge e is (*at most*) *l-crossed* if (at most) l distinct pseudolines intersect e in its interior. A 0-anchored 0-crossed non-aligned edge is also called *free*. A non-empty edge set $A \subset E$ is l-crossed if l is the smallest number such that every edge in A is at most l-crossed.

The *alignment complexity* of an aligned graph describes how "complex" the relationship between the graph G and the pseudoline arrangement $C_1, \ldots, C_k$ is. It is formally defined as a triple (l_0, l_1, l_2), where l_i, $i = 0, 1, 2$, indicates that E_i is at most l_i-crossed or has to be empty, if $l_i = \perp$. For example, an aligned graph where every

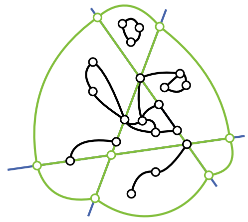


Figure 10.3: The black edges and vertices and the blue pseudoline arrangement is the input graph $(G, \mathcal{A})$. The green and black graph together depict the modified graph before the triangulation step.

vertex is aligned and every edge has at most l interior intersections has the alignment complexity $(\perp, \perp, l)$. For further examples we refer to Figure 10.2.

Theorem 10.1. *Every k-aligned graph $(G, \mathcal{A})$ of alignment complexity $(0, 0, 0)$ with a stretchable pseudoline arrangement $\mathcal{A}$ has an aligned drawing.*

Proof. We modify the graph $(G, \mathcal{A})$ as follows; see Figure 10.3. We place a vertex on each intersection of two or more pseudolines (if the intersection is not already occupied). In case that k is at least two, every unbounded cell C of $\mathcal{A}$ has two pseudosegments of infinite length. We place a vertex on each of them at infinity and connect them by an edge routed through the interior of C.

Further, let u and v be two C-aligned vertices, that are consecutive along C. If uv is not already an edge of G, we insert it into G and route it on C. Note that, since (G, C) does not contain edges that cross a pseudoline, the resulting graph is again an aligned graph of alignment complexity $(0, 0, 0)$. The boundary of every cell is covered by aligned edges. Thus, we can triangulate $(G, \mathcal{A})$ without introducing intersections between edges and a pseudoline.

We obtain an aligned drawing of the modified graph as follows. Note that the only interaction between two cells are the aligned vertices and edges on their common boundary, i.e., there are no edges crossing the boundary. Hence, for every pseudosegments of $\mathcal{A}$ we place the aligned vertices on it, arbitrarily (but respecting their order) on the corresponding line segment in A. Since, every cell is covered by aligned edges, we can draw the interior of two cells independently from each other. More formally, the vertex placements of the vertices of the pseudolines prescribes a convex drawing of the outer face of the graph G_C, i.e., the graph induced by the vertices in the interior or on the boundary of a cell C. Thus, we obtain a drawing Γ of G by applying the result of Tutte [Tut63] to each graph G_C, independently. □

We prove that the 2-aligned graph in Figure 10.4a does not have an aligned drawing.

Theorem 10.2. *There is a 2-aligned graph of alignment complexity $(\perp, 1, \perp)$ that does not have an aligned drawing.*

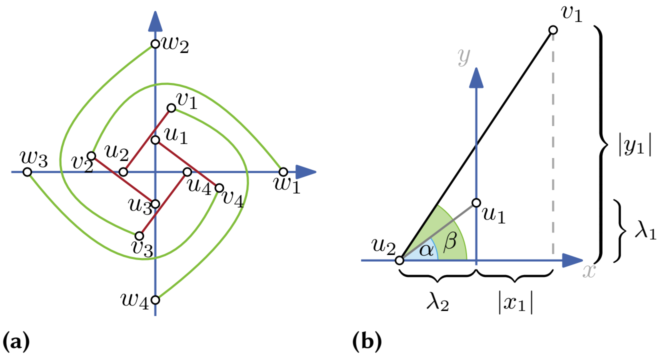


Figure 10.4: (a) A 2-aligned graph that does not have an aligned drawing. (b) We have $\lambda_1/\lambda_2 = \tan(\alpha) < \tan(\beta) = |y_1|/(\lambda_2 + |x_1|)$.

Proof. Assume that the aligned graph in Figure 10.4a has an aligned drawing. For $i = 1, \ldots, 4$ let (x_i, y_i) be the point for v_i, let λ_i be the distance of u_i to the origin O and let $\lambda_5 = \lambda_1$. Since u_2v_1 intersects the y-axis above u_1, edge u_2v_1 has a steeper slope than the segment u_2u_1; see Figure 10.4b. We obtain $\lambda_1/\lambda_2 < |y_1|/(\lambda_2 + |x_1|)$ and therefore $|x_1| < \lambda_2/\lambda_1 \cdot |y_1|$. Analogously, we obtain

$$|x_i| < \frac{\lambda_{i+1}}{\lambda_i} \cdot |y_i|, i = 1,3 \qquad |y_i| < \frac{\lambda_{i+1}}{\lambda_i} \cdot |x_i|, i = 2,4. \tag{10.1}$$

Since $v_{i+1}w_i$, with $v_5 = v_1$, are embedded as straight lines, we further get estimation (2) that $|y_i| < |y_{i+1}|$ for $i = 1,3$ and $|x_i| < |x_{i+1}|$ for $i = 2,4$ and $x_5 = x_1$. By multiplying the left and the right sides we obtain $|x_1|\cdot|y_2|\cdot|x_3|\cdot|y_4| \overset{(10.1)}{<} |y_1|\cdot|x_2|\cdot|y_3|\cdot|x_4|\cdot\frac{\lambda_2\lambda_3\lambda_4\lambda_1}{\lambda_1\lambda_2\lambda_3\lambda_4} = |y_1|\cdot|x_2|\cdot|y_3|\cdot|x_4| \overset{(2)}{<} |y_2|\cdot|x_3|\cdot|y_4|\cdot|x_1|$. A contradiction. □

10.3 Complexity and Fixed-Parameter Tractability

In this section, we deal with the topological setting where we are given a planar embedded graph $G = (V, E)$ and a subset $S \subseteq V$. We ask for a straight-line drawing of G where the vertices in S are collinear. According to Da Lozzo et al. [Da +18], this problem is equivalent to deciding the existence of a pseudoline C with respect to G passing exactly through the vertices in S. We refer to this problem as *pseudoline existence problem* and the corresponding search problem is referred to as *pseudoline construction problem.* Using techniques similar to Fößmeier and Kaufmann [FK97], we can show that the pseudoline existence problem is $\mathcal{NP}$-hard.

Let $G^\star + V$ be the graph obtained from the dual graph $G^\star = (V^\star, E^\star)$ of $G = (V, E)$ by placing every vertex $v \in V$ in its dual face $v^\star$ and connecting it to every vertex on the boundary of the face $v^\star$.

Lemma 10.3. *Let $G = (V, E)$ be a 3-connected 3-regular planar graph. There exists a pseudoline through V with respect to the graph $G^\star + V$ if and only if G is Hamiltonian.*

Proof. Recall that the dual of a 3-connected 3-regular graph is a triangulation with a single combinatorial embedding.

Assume that there exists a pseudoline $\mathcal{C}$ through V with respect to $G^\star + V$. Then the order of appearance of the vertices of $G^\star + V$ on $\mathcal{C}$ defines a sequence of adjacent faces in $G^\star$, i.e., vertices of the primal graph G that are connected via primal edges. This yields a Hamiltonian cycle in G.

Let C be a Hamiltonian cycle of G and consider a simultaneous embedding of G and $G^\star + V$ on the plane, where each pair of a primal and its dual edge intersects exactly once. Thus, the cycle C crosses each dual edge e at most once and passes through exactly the vertices V. There is a vertex v on the cycle C such that v lies in the unbounded face of $G^\star + V$. Thus, the cycle C can be interpreted as a pseudoline $\mathcal{C}(V)$ in $G^\star + V$ through all vertices in V by splitting it in the unbounded face of $G^\star + V$. □

Since computing a Hamiltonian cycle in 3-connected 3-regular planar graphs is $\mathcal{NP}$-complete [GJT76], we get that the pseudoline construction problem is $\mathcal{NP}$-hard. On the other hand, we can guess a sequence of vertices, edges and faces of G, and then test in polynomial time whether this corresponds to a pseudoline $\mathcal{C}$ with respect to G that traverses exactly the vertices in S. Thus, the pseudoline construction problem is in $\mathcal{NP}$. This proves the following theorem.

Theorem 10.4. *The pseudoline existence problem is $\mathcal{NP}$-complete.*

In the following, we show that the pseudoline construction problem is fixed-parameter tractable with respect to $|S|$. To this end, we construct a graph $G^{\text{tr}} = (V^{\text{tr}}, E^{\text{tr}})$ and a set $S^{\text{tr}} \subseteq V^{\text{tr}}$ with $|S^{\text{tr}}| \leq |S| + 1$ such that G^{tr} contains a simple cycle traversing all vertices in S^{tr} if and only if there exists a pseudoline $\mathcal{C}$ that passes exactly through the vertices in S such that $(G, \mathcal{C})$ is an aligned graph.

We observe that if the vertices S of a positive instance are not independent, they can only induce a *linear forest*, i.e., a set of paths, as otherwise, there is no pseudoline through all the vertices in S with respect to G. We call the edges on the induced paths *aligned edges*. An edge that is not incident to a vertex in S is called *crossable*, in the sense that only crossable edges can be crossed by $\mathcal{C}$, otherwise $\mathcal{C}$ is not a pseudoline with respect to G. Let $S_{\text{ep}} \subseteq S$ be the subset of vertices that are endpoints of the paths induced by S (an isolated vertex is a path of length 0). We construct G^{tr} in several steps; refer to Figure 10.5.

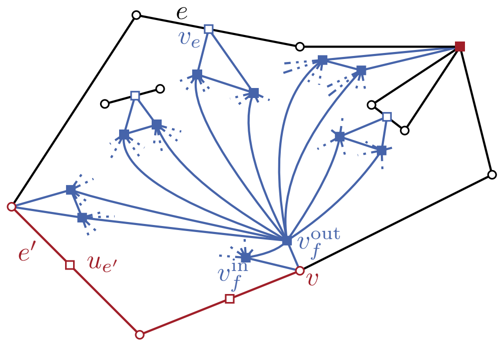


Figure 10.5: The black and red edges depict a single face of the input graph G. Red and blue edges build the transformed graph G^{tr}. Red round vertices are vertices in S, red squared vertices illustrate the set S^{tr}, the filled red square is a vertex in S and S^{tr}. Blue dashed edges sketch the clique edges between clique vertices (filled blue).

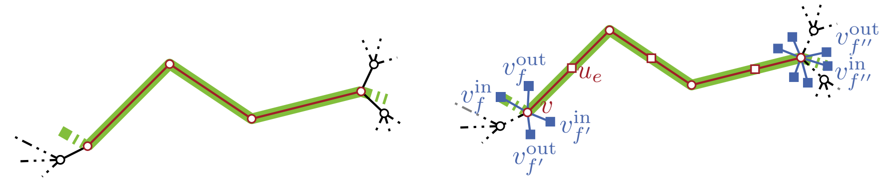
Step 1 Let G' be the graph obtained from G by subdividing each aligned edge e with a new vertex u_e and let S^{tr} be the set consisting of all isolated vertices in S and the new subdivision vertices. Additionally, we add to G' one new vertex o that we embed in the outer face of G and also add to S^{tr}. Observe that by construction $|S^{\text{tr}}| \le |S| + 1$. Finally, subdivide each crossable edge e by a new vertex v_e. We call these vertices *traversal nodes* and denote their set by $T = S_{\text{ep}} \cup \{v_e \mid e \text{ is crossable}\} \cup \{o\}$. Intuitively, a curve will correspond to a path that uses the vertices in S_{ep} to hop onto paths of aligned edges and the subdivision vertices of crossable edges to traverse from one face to another. Moreover, the vertex $o \in S^{\text{tr}}$ plays a similar role, forcing the curve to visit the outer face.

Step 2 For each face f of G' we perform the following construction. Let $T(f)$ denote the traversal nodes that are incident to f. For each vertex $v \in T(f)$ we create two new vertices v_f^{in} and v_f^{out}, add the edges vv_f^{in} and vv_f^{out} to G', and draw them in the interior of f. Finally, we create a clique $C(f)$ on the vertex set $\{v_f^{\text{in}}, v_f^{\text{out}} \mid v \in T(f)\}$, and embed its edges in the interior of f.

Step 3 To obtain G^{tr} remove all edges of G' that correspond to edges of G except those that stem from subdividing an aligned edge of G.

Lemma 10.5. *There exists a pseudoline C traversing exactly the vertices in S such that (G, C) is an aligned graph if and only if there exists a simple cycle in G^{tr} that traverses all vertices in S^{tr}.*

Proof. Suppose C is a cycle in G^{tr} that visits all vertices in S^{tr}. Without loss of generality, we assume that there is no face f such that C contains a subpath from v_f^{in} via v to v_f^{out}



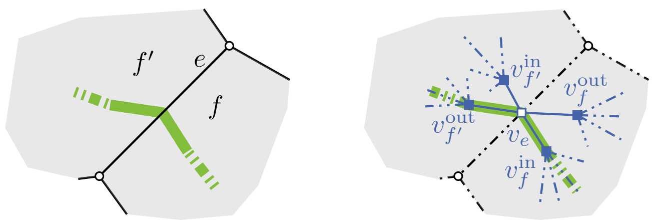
(a) **(b)**

Figure 10.6: (a) A pseudoline (thick green) traversing a path of aligned edges (thin red). (b) A path (thick green) in G^{tr} visiting consecutive vertices in S^{tr} (red squared).

(or its reverse) for some vertex $v \in T(f) \setminus S_{\mathrm{ep}}$, as otherwise we simply shortcut this path by the edge $v_f^{\mathrm{in}} v_f^{\mathrm{out}} \in C(f)$.

Consider a path P of aligned edges in G that contains at least one edge; refer to Figure 10.6. By definition, C visits all the subdivision vertices $u_e \in S^{\mathrm{tr}}$ of the edges of P, and thus it enters P on an endpoint of P, traverses P and leaves P at the other endpoint. All isolated vertices of S are contained in S^{tr}, and therefore C indeed traverses all vertices in S (and thus also all aligned edges). As described above, G^{tr} is indeed a topological graph, and thus C corresponds to a closed curve ρ that traverses exactly the vertices in S and the aligned edges.

We now show that ρ can be transformed to a pseudoline with respect to G. Let e be a non-aligned edge of G that has a common point with ρ in its interior; see Figure 10.7. Thus, C contains the subdivision vertex v_e. In particular, this implies that e is crossable. Moreover, from our assumption on C, it follows that C enters v_e via v_f^{in} or v_f^{out} and leaves it via $v_{f'}^{\mathrm{in}}$ or $v_{f'}^{\mathrm{out}}$, where f and f' are the faces incident to e, and it is $f \neq f'$ as we could shortcut C otherwise. Therefore, ρ indeed intersects e and uses it to traverse to a different face of G. Moreover, since e has only a single subdivision vertex in G^{tr} and C is simple, it follows that e is intersected only once. Thus ρ is a curve that intersects



(a) **(b)**

Figure 10.7: (a) A pseudoline (thick green) passing through a non-aligned edge. (b) A path (thick green) in G^{tr} traversing a subdivision vertex v_e (blue non-filled square). Black (dashed) segments are edges of G.

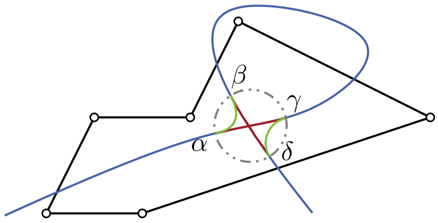


Figure 10.8: Resolving an intersection by exchanging the intersecting segments (red) with non-intersecting segments (green).

all vertices in S, traverses all aligned edges, and crosses each edge of G (including the endpoints) at most once. Moreover, ρ traverses the outer face since C contains o.

The only reasons why ρ is not necessarily a pseudoline with respect to G are that it is a closed curve and it may cross itself. However, we can break ρ in the outer face and route both ends to infinity, and remove such self-intersections locally as follows; see Figure 10.8. Consider a circle D around an intersection I that neither contains a second self-intersection nor a vertex, nor an edge of G. Let $\alpha, \beta, \gamma, \delta$ be the intersections of D with $\mathcal{C}$. We replace the pseudosegment $\alpha\gamma$ with a pseudosegment $\alpha\beta$, and $\beta\delta$ with a pseudosegment $\gamma\delta$. We route the pseudosegments $\alpha\beta$ and $\gamma\delta$ through the interior of D such that they do not intersect. Thus, we obtain a pseudoline $\mathcal{C}$ with respect to G that contains exactly the vertices in S.

For the converse assume that $\mathcal{C}$ is a pseudoline that traverses exactly the vertices in S such that $(G, \mathcal{C})$ is an aligned graph. The pseudoline $\mathcal{C}$ can be split into three parts $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ such that $\mathcal{C}_1$ and $\mathcal{C}_3$ have infinite length and do not intersect with G, and $\mathcal{C}_2$ has its endpoints in the outer face of G. We transform $\mathcal{C}$ into a closed curve $\mathcal{C}'$ by removing $\mathcal{C}_1, \mathcal{C}_3$ and adding a new piece connecting the endpoints of $\mathcal{C}_2$ without intersecting $\mathcal{C}_2$ or G. Additionally, we choose an arbitrary direction for $\mathcal{C}'$ in order to determine an order of the crossed edges and vertices.

We show that G^{tr} contains a simple cycle traversing the vertices in S^{tr}. By definition $\mathcal{C}'$ consists of two different types of pieces; see Figure 10.6. The first type traverses a path of aligned edges between two vertices in S_{ep}. The other type traverses a face of G by entering and exiting it either via an edge or from a vertex in S_{ep}; see Figure 10.9. We show how to map these pieces to paths in G^{tr}; the cycle C is obtained by concatenating all these paths.

Each piece of the first type indeed corresponds directly to a path in G^{tr}; see Figure 10.6. Consider now a piece π of the second type traversing a face f; refer to Figure 10.9. The piece π enters f either from a vertex in S_{ep} or by crossing a crossable edge e. In either case, $T(f)$ contains a corresponding traversal node u. Likewise, $T(f)$ contains a traversal node v for the edge or vertex that $\mathcal{C}'$ intersects next. We map π to the path $uu_f^{\text{in}}v_f^{\text{out}}v$ in G^{tr}. By construction, paths corresponding to consecutive pieces of C' share a traversal node, and therefore concatenating all paths yields a cycle C in G^{tr}. Moreover, C is simple, since C' intersects each edge and each vertex at most once. Note that C contains at least one edge of the outer face (as C' traverses the outer face), and we modify C so that it also traverses the special vertex o.

It remains to show that C contains all vertices in S^{tr}. There are three types of vertices in S^{tr}; the subdivision vertices of aligned edges, the isolated vertices in S, and the special vertex o. The latter is in C by the last step of the construction. The isolated vertices in S are traversed by C' and contained in S_{ep}, and they are therefore visited also by C. Finally, the subdivision vertices of aligned edges are traversed by the paths corresponding to the first type of pieces, since C' traverses all aligned edges. □

Theorem 10.6 (Wahlström [Wah13])**.** *Given an n-vertex graph $G = (V, E)$ and a subset $S \subseteq V$, it can be tested in $O(2^{|S|}\text{poly}(n))$ time whether a simple cycle through the vertices in S exists. If affirmative the cycle can be reported within the same asymptotic time.*

Theorem 10.7. *The pseudoline construction problem is solvable in $O(2^{|S|}\text{poly}(n))$ time, where n is the number of vertices.*

Proof. Let $G = (V, E)$ with $S \subseteq V$ be an instance of the pseudoline construction problem. By Lemma 10.5 the pseudoline construction problem is equivalent to determining whether G^{tr} contains a simple cycle visiting all vertices in S^{tr}. Since the size of G^{tr} is $O(n^2)$ and it can be constructed in $O(n^2)$ time, and $|S^{\text{tr}}| \leq |S| + 1$, Theorem 10.6 can be used to solve the latter problem in the desired running time. □

We note that indeed the construction of G^{tr} only allows leaving a path of aligned edges at an endpoint in S_{ep}. Therefore, a single vertex in S^{tr} for each path of aligned edges would be sufficient to ensure that C traverses the whole path. Thus, by removing for each path all but one vertex from S^{tr} we obtain an algorithm that is FPT with respect to the number of paths induced by S.

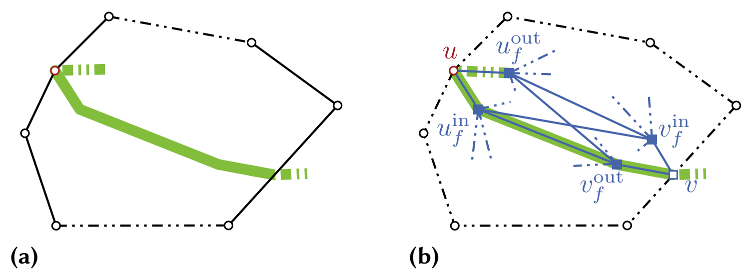


Figure 10.9: (a) A pseudoline piece π (thick green) passing through a face f. (b) Path (thick green) in G^{tr} corresponding to π.

Theorem 10.8. *The pseudoline construction problem is solvable in $O(2^P \text{poly}(n))$ time, where n is the number of vertices and P is the number of paths induced by the vertex set S to be aligned.*

10.4 Drawing Aligned Graphs

We show that every aligned graph where each edge either entirely lies on a pseudoline or is intersected by at most one pseudoline, i.e., alignment complexity $(1, 0, \perp)$, has an aligned drawing. For 1-aligned graphs we show the stronger statement that every 1-aligned graph has an aligned drawing with a given aligned convex drawing of the outer face. We first present our proof strategy and then deal with 1- and k-aligned graphs.

10.4.1 Proof Strategy

Our general strategy for proving the existence of aligned drawings of an aligned graph $(G, \mathcal{A})$ is as follows. First, we show that we can triangulate $(G, \mathcal{A})$ by adding vertices and edges without invalidating its properties. We can thus assume that our aligned graph $(G, \mathcal{A})$ is an aligned triangulation. Second, we show that unless G has a specific structure (e.g., a k-wheel or a triangle), it contains an aligned or a free edge. Third, we exploit the existence of such an edge to reduce the instance. Depending on whether the edge is contained in a separating triangle or not, we either decompose along that triangle or contract the edge. In both cases the problem reduces to smaller instances that are almost independent. In order to combine solutions, it is, however, crucial to use the same arrangement of lines A for both of them.

In the following, we introduce the necessary tools used for all three steps on k-aligned graphs of alignment complexity $(1, 0, \perp)$. Recall, that for this class (i) every non-aligned edge is at most 1-crossed, (ii) every 1-anchored edge is 0-crossed, and (iii) there is no edge with its endpoints on two pseudolines.

Lemmas 10.9 – 10.12 show that every aligned graph of alignment complexity $(1, 0, \perp)$ has an aligned triangulation with the same alignment complexity. If G contains a separating triangle, Lemma 10.13 shows that $(G, \mathcal{A})$ admits an aligned drawing if both split components have an aligned drawing. Finally, with Lemma 10.14 we obtain a drawing of $(G, \mathcal{A})$ from a drawing of the aligned graph $(G/e, \mathcal{A})$ where one particular edge e is contracted.

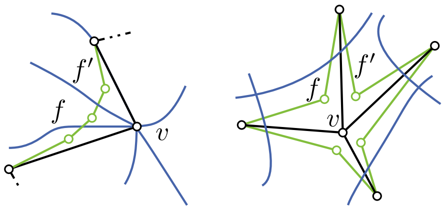
Lemma 10.9. *Let $(G, \mathcal{A})$ be a k-aligned n-vertex graph of alignment complexity $(1, 0, \perp)$. Then there exists a biconnected k-aligned graph $(G', \mathcal{A})$ that contains G as a subgraph. The set $E(G') \setminus E(G)$ has alignment complexity $(1, 0, \perp)$ and does not contain aligned edges. The size of $E(G') \setminus E(G)$ is in $O(nk + k^3)$.*

Proof. Our procedure works in two steps. First, we connect disconnected components. Second, we assure that the graph is biconnected by inserting edges around a cut-vertex. Initially, we place a vertex in every cell that does not contain a vertex in its interior.

Consider a cell C of $\mathcal{A}$ that contains two vertices u and v that belong to distinct connected components G_u and G_v. We refer to two vertices u, v that lie in the interior or on the boundary of C as *C-visible* if there is a curve in the interior of C that connects u to v and that does not intersect G except at its endpoints. In the following, we exhaustively connect C-visible pairs of vertices of distinct connected components of G. If u and v are C-visible, we simply connect them by an edge e. In case that both vertices are aligned, we have to subdivide the edge e with a vertex to avoid introducing 2-anchored edges to the graph. Assume that u, v are not C-visible. Consider any curve ρ in the interior of C that connects u and v. Then ρ intersects a set of edges of G either in their interior or in a vertex. Thus, there are two edges e_1 and e_2 consecutive along ρ, that belong two distinct connected components. Since e_1 and e_2 are at most 1-crossed, there is an endpoint of e_1 and an endpoint of e_2 that are C-visible and thus can be connected by an edge. Overall it is sufficient to add a linear number of edges to join distinct connected components that have vertices in a common cell.

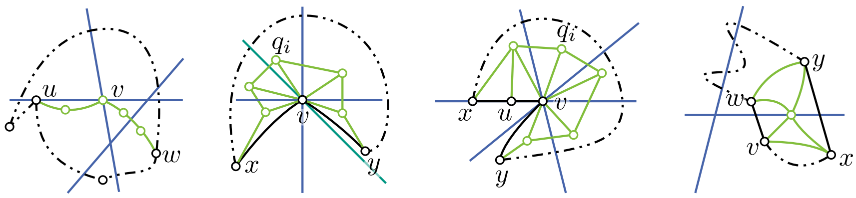
By construction, every cell contains at least one free vertex. Thus, in order to connect the graph we consider two cells C_1, C_2 with a common boundary. Assume that there is a vertex u on the common boundary. In this case, the previous step ensures that there is a path from u to every vertex that lies in the interior or on the boundary of C_1 or C_2. Hence, consider the case where no vertex lies on the common boundary of the two cells. Moreover, the common boundary does also not contain an edge, since this edge would be 2-anchored or l-crossed, $l \geq 2$. Similar to the previous step, we can connect two arbitrary vertices of C_1 and C_2 with a curve ρ that intersects the common boundary. If this curve does not intersect an edge we can simply connect the two vertices with an edge. Otherwise, at least in one cell $C' \in \{C_1, C_2\}$ the curve intersects at least one edge. Therefore, there is an edge e' that comes immediately before the intersection of ρ with the boundary of C'. Since every edge is at most 1-crossed, there are two vertices in C_1 and C_2 that can be connected by an edge. Due to the previous step, we can assume that the vertices in the interior of each cell are connected by a path. Thus, we add at most one edge for each pair of adjacent cells. Since there are $O(k^2)$ cells we add $O(k^2)$ vertices and edges to G, i.e., the size of G is $O(n + k^2)$.

We now assume that G is connected but not biconnected and has $n' \in O(n + k^2)$ vertices. Consider a single cut vertex v; refer to Figure 10.10. We consider the common arrangement $\mathcal{F}$ of $\mathcal{A}$ and G, i.e., a face can be restricted by pseudosegments of $\mathcal{A}$ and edges of G. Let $\mathcal{F}_v$ be the set of faces in $\mathcal{F}$ with v on their boundary. We place a vertex v_f in every face f of $\mathcal{F}_v$. Let f and f' be two distinct faces of $\mathcal{F}_v$ with a common edge ϵ on their boundary. If ϵ is an edge uv of G, we insert the edges uv_f and $uv_{f'}$. Since



(a) **(b)**

Figure 10.10: Green edges and vertices are added around a cut-vertex v to connect the connected components (black) incident to v. (a) v is an intersection vertex. (b) v is a free vertex.



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

Figure 10.11: Black lines indicate a face f of G. Light green edges or vertices are newly added into f. Blue lines denote the pseudoline arrangement. (a) Isolation of an intersection. (b-c) Isolation of an aligned vertex or edge. (d) Isolation of a pseudosegment.

uv is at most 1-crossed, the new edges are as well at most 1-crossed. If ϵ corresponds to a pseudosegment, we insert the edge $v_f v_{f'}$ such that it crosses ϵ. Since v_f and $v_{f'}$ are free vertices, the edge is by construction 1-crossed.

This procedure adds $O(k + \deg v)$ vertices and edges around v, since at most k pseudolines intersect in a single point. The degree of vertices adjacent to v is increased by at most 2. Thus, the size of G increases to $O(n'k)$. Thus, we have that the size of G is $O(nk + k^3)$. □

Lemma 10.10. *Let $(G, \mathcal{A})$ be a biconnected k-aligned n-vertex graph of alignment complexity $(1, 0, \perp)$. There exists a k-aligned triangulation $(G_T = (V_T, E_T), \mathcal{A})$ of f whose size is $O(nk + k^3)$. The set $E(G_T) \setminus E(G)$ has alignment complexity $(1, 0, \perp)$ and does not contain aligned edges.*

Proof. We call a face *non-triangular* if its boundary contains more than three vertices. An aligned vertex v or an aligned edge e is *isolated* if all faces with v or e on their boundaries are triangles. A pseudosegment s is *isolated* if s does not intersect the interior of a simple cycle. Our proof distinguishes four cases. Each case is applied exhaustively in this order.

1. If the interior of f contains the intersection of two or more pseudolines, we split the face so that there is a vertex that lies on the intersection.
2. If the boundary of a face has an aligned vertex or an aligned edge, we isolate the vertex or the edge from f.
3. If the interior of a face f intersects a pseudoline C, then it subdivides C into a set of pseudosegments. We isolate each of the pseudosegments independently.
4. Finally, if none of the previous cases apply, i.e., neither the boundary nor the interior of f contains parts of a pseudoline, the face f can be triangulated with a set of additional free edges.

Let $\mathcal{A}_f$ be the arrangement of $\mathcal{A}$ restricted to the interior of f.

1. Let f be a non-triangular face whose interior contains an intersection of two or more pseudolines; see Figure 10.11a. We place a vertex on every intersection in the interior of f. We obtain a biconnected graph G_1 with the application of Lemma 10.9. Since there are $O(k^2)$ intersections, the size of G_1 is $O\left((n + k^2)k + k^3\right) = O(nk + k^3)$.
2. Let f_1 be a non-triangular face of G_1 with an aligned vertex or an aligned edge uv on its boundary. Further, the interior of f_1 does not contain the intersection of a set of pseudolines; see Figure 10.11b and 10.11c. In case of an aligned vertex we simply assume $u = v$. Since G is biconnected, there exist two edges xu, vy on the boundary of f_1. Let $C_1, \dots, C_l \in \text{cells}(\mathcal{A}_{f_1})$ be cells with u or v on their boundary, such that C_i is adjacent to C_{i+1}, $i < l$. Since f_1 does not contain 2-anchored edges, at most one of the vertices u and v can be an intersection vertex. Thus, l is at most $2k$. We construct an aligned graph $(G_2, \mathcal{A})$ from $(G_1, \mathcal{A})$ as follows. We place a vertex q_i in the interior of each cell C_i, $i \le l$. Let $q_0 = x$ and $q_{l+1} = y$. We insert edges $e_i = q_i q_{i+1}$, $i = 0, \dots, l$ in the interior of f_1 so that the interior of e_i crosses the common boundary of C_i and C_{i+1} exactly once and it crosses no other boundary. Thus, if the edge e_i is either incident to x or to y, it at most 1-anchored and 0-crossed. Otherwise, it is 0-anchored and 1-crossed. The added path splits f into two faces f', f'' with a unique face f' containing u and v on its boundary. If $w \in \{u, v\}$ is on the boundary of cell C_i, we insert an edge wq_i. Each edge wq_i is 1-anchored and 0-crossed. Let C_i and C_{i+1} be two cells incident to w. Then, the vertices w, q_i, q_{i+1} form a triangle. If $u \neq v$, there is a unique cell C_i incident to u and v. Hence, the vertices u, v, q_i form a triangle. Moreover, for $1 \le i \le l$, every edge uq_i and vq_i is incident to two triangles. Therefore, f' is triangulated. By construction, we do not insert aligned vertices and edges, thus the number of aligned edges and aligned vertices of f'' is one less compared to f_1. Hence, we can inductively proceed on f''.
   Assume the aligned vertex v is an intersection vertex. Thus, isolating v uses $O(k)$ additional vertices and edges. Therefore, all intersection vertices can be isolated with $O(k^3)$ vertices and edges.

Now consider an aligned vertex v that is not an intersection vertex. In this case v is incident to at most two cells. We can isolate all such vertices with $O(n)$ vertices and edges. The same bound holds for aligned edges. Finally, we obtain an aligned graph $(G_2, \mathcal{A})$ of size $O(nk + k^3)$.

3. Let f_2 be a non-triangular face of G_2 whose interior intersects a pseudoline C and has no aligned edge and no aligned vertex on its boundary. Further, the interior of f_2 does not contain the intersection of two or more pseudolines. Then the face f_2 subdivides C into a set of pseudosegments; see Figure 10.11d. We iteratively isolate such a pseudosegment $\mathcal{S}$. Since f_2 does not contain the intersection of two or more pseudolines in its interior, there are two distinct cells $C_1 \in \text{cells}(\mathcal{A}_f)$ and $C_2 \in \text{cells}(\mathcal{A}_f)$ with $\mathcal{S}$ on their boundary. Since f_1 neither contains an aligned vertex nor an aligned edge and G is biconnected, there are exactly two edges $e_1 = vw$ and $e_2 = xy$ with the endpoints of $\mathcal{S}$ in the interior of these edges and v, x and w, y on the boundaries of C_1 and C_2, respectively. Since f_2 does not have an l-crossed edge, $l \geq 2$, and every 1-crossed edge is 0-anchored, the vertices v, w, x, y are free. We construct a graph G' by placing a vertex u on s and inserting edges uv, uw, ux uy, vx and wy. We route each edge so that the interior of an edge does not intersect the boundary of a cell $C_i, i = 1, 2$. Thus, the edges vx and wy are free and the others are 1-anchored and 0-crossed.
   Every edge in G_2 is at most 1-crossed, thus the number of pseudosegments is linear in the size of G_2. Therefore, we add a number of vertices and edges that is linear in the size of G_2.
   Thus, we obtain an aligned graph $(G_3, \mathcal{A})$ of size $O(nk + k^3)$.
4. If none of the cases above applies to a non-triangular face f_4 of G_3, then neither the interior nor the boundary of the face intersects a pseudoline C_i. Thus, we can triangulate f_4 with a number of free edges linear in the size of f_4. Thus, in total we obtain an aligned triangulation $(G_T, \mathcal{A})$ of $(G, \mathcal{A})$ of size $O(nk + k^3)$.

□

Observe that the correctness of the previous triangulation procedure only relies on the fact that every non-triangular face contains at most 1-crossed edges. While Lemma 10.10 is sufficient for our purposes, for the sake of generality, we show how to isolate l-crossed edges. This allows us to triangulate biconnected aligned graphs without increasing the alignment complexity.

Theorem 10.11. *Every biconnected k-aligned n-vertex graph $(G, \mathcal{A})$ of alignment complexity (l_0, l_1, l_2) has an aligned triangulation $(G_T, \mathcal{A})$. The alignment complexity of $E(G_T) \setminus E(G)$ is $(\max\{l_0, 1\}, l_1, l_2)$ and the size of this set is $O(nk + k^3)$.*

Proof. For $l \geq 1$, we iteratively isolate l-crossed edges uv from a non-triangular face f as sketched in Figure 10.12. Let $C_0, C_1, \ldots, C_l \in \text{cells}(\mathcal{A})$ be the cells in f that occur

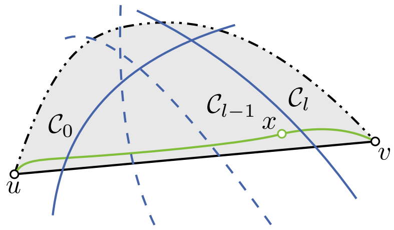


Figure 10.12: An l-crossed edge uv in a (grey) face f and a pseudoline arrangement (blue). The green edges isolate the edge uv.

in this order along uv. If one of these vertices is free, say v, we place a new vertex x in the interior of C_{l-1}. We insert the two edges ux, xv and route both edges close to uv. This isolates the edge uv from f. Notice that the edge xv is 0-anchored and 1-crossed and the edge ux $(l-1)$-crossed. In case that $l_0 \geq 1$, the alignment complexity of the new aligned graph is (l_0, l_1, l_2). Otherwise, the alignment complexity is $(1, l_1, l_2)$. If u and v are aligned, we place x on the boundary of C_{l-1} and C_l and route the edges ux and vx as before. The alignment complexity is not affected by this operation. The face uvx is triangular and therefore the edge uv is processed as above at most twice.

This procedure introduces a new $(l-1)$-crossed edge. Repeating the process $l-2$ times generates a new face f' from f where edge uv is substituted by a path of at most 1-crossed edges. To isolate all l-crossed edges in $(G, \mathcal{A})$, we add $O(kn)$ vertices and edges.

By isolating all l-crossed edges in this way, we obtain an aligned graph where every non-triangular face is bounded by at most 1-crossed edges. The proof of Lemma 10.10 handles all non-triangular faces independently. For the correctness of the triangulation it is sufficient to ensure that every non-triangular face does neither contain 2-anchored edges nor l-crossed edges. Thus, we can apply the methods used in the proof of Lemma 10.10 to triangulate $(G, \mathcal{A})$ with $O(nk + k^3)$ additional vertices and edges. □

We now return to the treatment of aligned graphs with alignment complexity $(1, 0, \perp)$. To simplify the proofs, we augment the input graph with an additional cycle in the outer face that contains all intersections of $\mathcal{A}$ in its interior, and we add subdivision vertices on the intersections of C_i-aligned edges with pseudolines C_j, $i \neq j$. A k-aligned graph is *proper* if (i) every aligned edge is 0-crossed, (ii) for $k \geq 2$, every edge on the outer face is 1-crossed, (iii) the boundary of the outer face intersects every pseudoline exactly twice, and (iv) the outer face does not contain any intersection of $\mathcal{A}$.

An aligned graph $(G_{\text{rs}}, \mathcal{A})$ is a *rigid subdivision* of an aligned graph $(G, \mathcal{A})$ if and only if G_{rs} is a subdivision of G and every subdivision vertex is an intersection vertex with respect to $\mathcal{A}$. We show that we can extend every k-aligned graph $(G, \mathcal{A})$ to a proper k-aligned triangulation.

Lemma 10.12. *For every $k \geq 2$ and every k-aligned n-vertex graph $(G, \mathcal{A})$ of alignment complexity $(1, 0, \perp)$, let $(G_{\mathrm{rs}}, \mathcal{A})$ be a rigid subdivision of $(G, \mathcal{A})$. Then there exists a proper k-aligned triangulation $(G', \mathcal{A})$ of alignment complexity $(1, 0, \perp)$ such that G_{rs} is a subgraph of G'. The size of G' is in $O(nk^2 + k^4)$. The set $E(G') \setminus E(G_{\mathrm{rs}})$ has alignment complexity $(1, 0, \perp)$ and does not contain aligned edges.*

Proof. We construct a rigid subdivision $(G_{\mathrm{rs}}, \mathcal{A})$ from $(G, \mathcal{A})$ by placing subdivision vertices on the intersections of C_i-aligned edges with pseudolines $C_j, i \neq j$. The number n_{rs} of vertices of G_{rs} is in $O(n + k^2)$.

We obtain a proper biconnected k-aligned graph $(G_b, \mathcal{A})$ by embedding a simple cycle C in the outer face of G_{rs} and applying Lemma 10.9. In order to construct C, we place a vertex v_c in each unbounded cell c of $\mathcal{A}$ and connect two vertices v_c and $v_{c'}$ if the boundaries of the cells c and c' intersect. The size n_b of G_b is $O(n_{\mathrm{rs}}k + k^3) = O(nk + k^3)$. We obtain a proper k-aligned triangulation $(G', \mathcal{A})$ of G_b with the application of Lemma 10.10. The size n' of G' is in $O(n_b k + k^3) = O((nk + k^3)k + k^3) = O(nk^2 + k^4)$. □

The following two lemmas show that we can reduce the size of the aligned graph and obtain a drawing by merging two drawings or by geometrically uncontracting an edge.

Lemma 10.13. *Let $(G, \mathcal{A})$ be a k-aligned triangulation. Let T be a separating triangle splitting G into subgraphs $G_{\mathrm{in}}, G_{\mathrm{out}}$ so that $G_{\mathrm{in}} \cap G_{\mathrm{out}} = T$ and G_{out} contains the outer face of G. Then, (i) $(G_{\mathrm{out}}, \mathcal{A})$ and $(G_{\mathrm{in}}, \mathcal{A})$ are k-aligned triangulations, and (ii) $(G, \mathcal{A})$ has an aligned drawing if and only if there exists a common line arrangement A such that $(G_{\mathrm{out}}, \mathcal{A})$ has an aligned drawing $(\Gamma_{\mathrm{out}}, A)$ and $(G_{\mathrm{in}}, \mathcal{A})$ has an aligned drawing $(\Gamma_{\mathrm{in}}, A)$ with the outer face drawn as $\Gamma_{\mathrm{out}}[T]$.*

Proof. It is easy to verify that $(G_{\mathrm{out}}, \mathcal{A})$ and $(G_{\mathrm{in}}, \mathcal{A})$ are aligned triangulations. An aligned drawing (Γ, A) of $(G, \mathcal{A})$ immediately implies the existence of an aligned drawing $(\Gamma_{\mathrm{out}}, A)$ of $(G_{\mathrm{out}}, \mathcal{A})$ and $(\Gamma_{\mathrm{in}}, A)$ of $(G_{\mathrm{in}}, \mathcal{A})$.

Let $(\Gamma_{\mathrm{out}}, A)$ be an aligned drawing of $(G_{\mathrm{out}}, \mathcal{A})$. Since $(\Gamma_{\mathrm{out}}, A)$ is an aligned drawing, $(\Gamma_{\mathrm{out}}[T], A)$ is an aligned drawing of $(T, \mathcal{A})$. Let $(\Gamma_{\mathrm{in}}, A)$ be an aligned drawing of $(G_{\mathrm{in}}, \mathcal{A})$ with the outer face drawn as $\Gamma_{\mathrm{out}}[T]$. Let Γ be the drawings obtained by merging the drawing Γ_{out} and Γ_{in}. Since $(\Gamma_{\mathrm{out}}, A)$ and $(\Gamma_{\mathrm{in}}, A)$ are aligned drawings on the same line arrangement A, (Γ, A) is an aligned drawing of $(G, \mathcal{A})$. □

Let $e = uv$ be an edge of G and assume that v is contracted onto u in the graph G/e. For a fixed edge $e = uv$, let f_e a function that maps the edges of $E(G) \setminus \{u, v\}$ to $E(G/e)$ that maps an ed

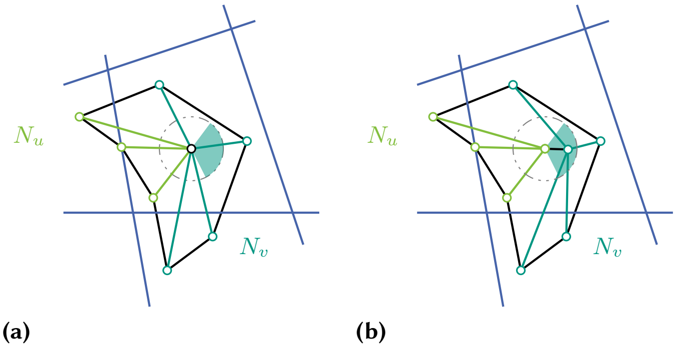


Figure 10.13: Unpacking an edge in a drawing Γ' of G/e (a) to obtain a drawing Γ of G (b).

Lemma 10.14. *Let $(G, \mathcal{A})$ be a proper k-aligned triangulation of alignment complexity $(1, 0, \perp)$ and let e be an interior 0-anchored aligned edge or an interior free edge of G that does not belong to a separating triangle and is not a chord. Then $(G/e, \mathcal{A})$ is a proper k-aligned triangulation of alignment complexity $(1, 0, \perp)$. Further, $(G, \mathcal{A})$ has an aligned drawing if $(G/e, \mathcal{A})$ has an aligned drawing.*

Proof. We first prove that $(G/e, \mathcal{A})$ is a proper k-aligned triangulation. Consider a topological drawing of the aligned graph $(G, \mathcal{A})$. Let c be the vertex in G/e obtained from contracting the edge $e = uv$. We place c at the position of u. Thus, all the edges incident to u keep their topological properties. We route the edges incident to v close to the edge uv within the cell from which they arrive to v in $(G, \mathcal{A})$. Since e is not an edge of a separating triangle, G/e is simple and triangulated.

Consider a free edge e. Observe that the triangular faces incident to e do not contain an intersection of two pseudolines in their interior, since $(G, \mathcal{A})$ does not contain l-crossed edges, for $l \geq 2$. Therefore, $(G/e, \mathcal{A})$ is an aligned triangulation. Since e is not a chord, $(G/e, \mathcal{A})$ is proper. Further, u and v lie in the interior of the same cell, thus, the edges incident to c have the same alignment complexity as in $(G, \mathcal{A})$.

If e is aligned, it is also 0-crossed, since $(G, \mathcal{A})$ is proper. Since e is also 0-anchored, the triangles incident to e do not contain an intersection of two pseudolines and therefore $(G/e, \mathcal{A})$ is a proper aligned triangulation. The routing of the edges incident to c, as described above, ensures that the alignment complexity is $(1, 0, \perp)$.

Let (Γ', A) be an aligned drawing of $(G/e, \mathcal{A})$. We now prove that $(G, \mathcal{A})$ has an aligned drawing. Let Γ'' denote the drawing obtained from Γ' by removing c together with its incident edges and let f denote the face of Γ'' where c used to lie. Since G/e is triangulated and e is an interior edge and not a chord, f is star-shaped and c lies inside the kernel of f; see Figure 10.13. We construct a drawing Γ of G as follows. If one of vertices u and v lies on the outer face, we assume, without loss of generality, that vertex to be u. First, we place u at the position of c and insert all edges incident to u. This results in a drawing of the face f' in which we have to place v. Since u is placed in the kernel of f, f' is star-shaped. If e is a free edge, the vertex v has to be placed in the same cell as u. We then place v inside f' sufficiently close to c so that it lies inside the kernel of f' and in the same cell as u. All edges incident to v are at most 1-crossed, thus, (Γ, A) is an aligned drawing of $(G, \mathcal{A})$.

Likewise, if e is an C-aligned edge, then v has to be placed on the line $L \in A$ corresponding to C. In this case, also c and therefore u lie on L. Since e is an interior edge, there exist two triangles uv, vx, xu and uv, vy, yu sharing the edge uv. Since, e is not part of a separating triangle, x and y are on different sides of L. Therefore the face f' contains a segment of the line L of positive length that is within the kernel of f'. Thus, we can place v close to u on the line L such that the resulting drawing is an aligned drawing of $(G, \mathcal{A})$. □

Note that contracting a 1-anchored aligned edge can result in a graph $(G/e, \mathcal{A})$ with an alignment complexity that does not coincide with the alignment complexity of $(G, \mathcal{A})$. Further, for general alignment complexities there is an aligned graph $(G, \mathcal{A})$ and an 1-anchored aligned edge e such that $(G/e, \mathcal{A})$ is not an aligned graph.

10.4.2 One Pseudoline

We show that every 1-aligned graph $(G, \mathcal{R})$ has an aligned drawing (Γ, R), where $\mathcal{R}$ is a single pseudoline and R is the corresponding straight line. Using the techniques from the previous section, we can assume that $(G, \mathcal{R})$ is a proper 1-aligned triangulation. We show that unless G is very small, it contains an edge with a certain property. This allows for an inductive proof to construct an aligned drawing of $(G, \mathcal{R})$.

Lemma 10.15. *Let $(G, \mathcal{R})$ be a proper 1-aligned triangulation without chords and with k vertices on the outer face. If G is neither a triangle nor a k-wheel whose center is aligned, then $(G, \mathcal{R})$ contains an interior aligned or an interior free edge.*

Proof. We first prove two useful claims.

Claim 1. Consider the order in which $\mathcal{R}$ intersects the vertices and edges of G. If vertices u and v are consecutive on $\mathcal{R}$, then the edge uv is in G and aligned.

Proof of the Claim. Observe that the edge uv can be inserted into G without creating crossings. Since G is a triangulation, it therefore already contains uv, and further, since every non-aligned edge has at most one of its endpoints on $\mathcal{R}$, it follows that indeed uv is aligned. This proves the claim. ◁

Claim 2. If $(G, \mathcal{R})$ is an aligned triangulation without aligned edges and x is an interior free vertex of G, then x is incident to a free edge.

Proof of the Claim. Assume for a contradiction that all neighbors of x lie either on $\mathcal{R}$ or on the other side of $\mathcal{R}$. First, we slightly modify $\mathcal{R}$ to a curve $\mathcal{R}'$ that does not

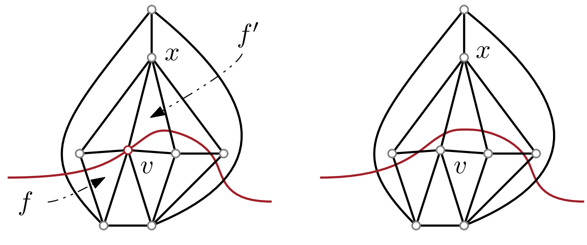


Figure 10.14: Transformation from a red vertex (a) to a gray vertex (b).

contain any vertices. Assume v is an aligned vertex; see Figure 10.14. Since there are no aligned edges, $\mathcal{R}$ enters v from a face f incident to v and leaves it to a different face f' incident to v. We then reroute $\mathcal{R}$ from f to f' locally around v. If v is incident to x, we choose the rerouting such that it crosses the edge vx.

Notice that if an edge e intersects $\mathcal{R}$ in its endpoints, then $\mathcal{R}'$ either does not intersect it or intersects it in an interior point. Moreover, e cannot intersect $\mathcal{R}'$ twice as in such a case $\mathcal{R}$ would pass through both its endpoints. Now, since G is a triangulation and the outer face of G is proper, $\mathcal{R}'$ corresponds to a simple cycle in the dual $G^\star$ of G, and hence corresponds to a cut C of G. Let H denote the connected component of $G - C$ that contains x and note that all edges of H are free. By the assumption and the construction of $\mathcal{R}'$, x is the only vertex in H. Thus, $\mathcal{R}'$ intersects only the faces incident to x, which are interior. This contradicts the assumption that $\mathcal{R}'$ passes through the outer face of G and finishes the proof of the claim. ◃

We now prove the lemma. Assume that G is neither a triangle nor a k-wheel whose center is aligned. If G is a k-wheel whose center is free, we find a free edge by Claim 2. Otherwise, G contains at least two interior vertices. If one of these vertices is free, we find a free edge by Claim 2. Otherwise, all interior vertices are aligned. Since G does not contain any chord, there is a pair of aligned vertices consecutive along $\mathcal{R}$. Thus by Claim 1 the instance $(G, \mathcal{R})$ has an aligned edge. □

Theorem 10.16. *Let $(G, \mathcal{R})$ be a proper aligned graph and let (Γ_O, R) be a convex aligned drawing of the aligned outer face $(O, \mathcal{R})$ of G. There exists an aligned drawing (Γ, R) of $(G, \mathcal{R})$ with the same line R and the outer face drawn as Γ_O.*

Proof. Given an arbitrary proper aligned graph $(G, \mathcal{R})$, we first complete it to a biconnected graph and then triangulate it by applying Lemma 10.9 and Lemma 10.10, respectively.

We prove the claim by induction on the size of G. If G is just a triangle, then clearly (Γ_O, R) is the desired drawing. If G is the k-wheel whose center is aligned, placing the vertex on the line in the interior of Γ_O yields an aligned drawing of G. This finishes the base case.

If G contains a chord e, then e splits $(G, \mathcal{R})$ into two graphs G_1, G_2 with $G_1 \cap G_2 = e$. It is easy to verify that $(G_i, \mathcal{R})$ is an aligned graph. Let (Γ_O^i, R) be a drawing of the face of $\Gamma_O \cup e$ whose interior contains G_i. By the inductive hypothesis, there exists an aligned drawing of (Γ_i, R) with the outer face drawn as (Γ_O^i, R). We obtain a drawing Γ by merging the drawings Γ_1 and Γ_2. The fact that both (Γ_1, R) and (Γ_2, R) are aligned drawings with a common line R and compatible outer faces implies that (Γ, R) is an aligned drawing of $(G, \mathcal{R})$.

If G contains a separating triangle T, let G_{in} and G_{out} be the respective split components with $G_{\text{in}} \cap G_{\text{out}} = T$. By Lemma 10.13, the graphs $(G_{\text{in}}, \mathcal{R})$ and $(G_{\text{out}}, \mathcal{R})$ are aligned graphs. By the induction hypothesis there exists an aligned drawing (Γ_{out}, R) of the aligned graphs $(G_{\text{out}}, \mathcal{R})$ with the outer face drawn as (Γ_O, R). Let $\Gamma[T]$ be the drawing of T in Γ_{out}. Further, $(G_{\text{in}}, \mathcal{R})$ has by induction hypothesis an aligned drawing with the outer face drawn as $\Gamma[T]$. Thus, by Lemma 10.13 we obtain an aligned drawing of $(G, \mathcal{R})$ with the outer face drawn as Γ_O.

If G is neither a triangle nor a k-wheel, by Lemma 10.15, it contains an interior aligned or an interior free edge e. Since e is not a chord and does not belong to a separating triangle, by Lemma 10.14, $(G/e, \mathcal{R})$ is an aligned graph and by the induction hypothesis it has an aligned drawing (Γ', R) with the outer face drawn as Γ_O. It thus follows by Lemma 10.14 again that $(G, \mathcal{R})$ has an aligned drawing with the outer face drawn as Γ_O. □

10.4.3 Alignment Complexity $(1, 0, \perp)$

We now consider k-aligned graphs $(G, \mathcal{A})$ of alignment complexity $(1, 0, \perp)$, i.e., every edge with two free endpoints intersects at most one pseudoline, every 1-anchored edge has no interior intersection with a pseudoline, and 2-anchored edges are entirely forbidden. In this section, we prove that every such k-aligned graph has an aligned drawing. As before we can assume that $(G, \mathcal{A})$ is a proper aligned triangulation. We show that if the structure of the graph is not sufficiently simple, it contains an edge with a special property. Further, we prove that every graph with a sufficiently simple structure indeed has an aligned drawing. Together this again enables an inductive proof that $(G, \mathcal{A})$ has an aligned drawing. Figure 10.15 illustrates the statement of the following lemma.

Lemma 10.17. *For $k \geq 2$ let $(G, \mathcal{A})$ be a proper k-aligned triangulation of alignment complexity $(1, 0, \perp)$ that neither contains a free edge, nor a 0-anchored aligned edge, nor a separating triangle. Then (i) every intersection contains a vertex, (ii) every cell of the pseudoline arrangement contains exactly one free vertex, (iii) every pseudosegment is either covered by two aligned edges or it intersects a single edge.*

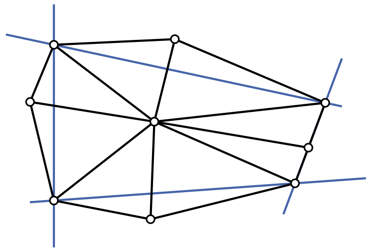


Figure 10.15: All possible variations of vertices and edges in Lemma 10.17.

Proof. The statement follows from the following sequence of claims. We refer to an aligned vertex that is not an intersection vertex as a *flexible aligned* vertex.

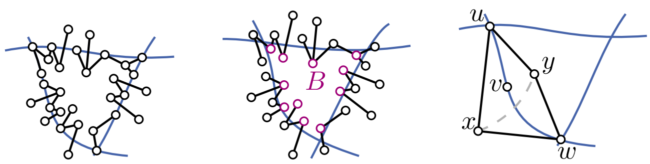
Claim 1. Every intersection contains a vertex.

Assume that there is an intersection I that does not contain a vertex. Since $(G, \mathcal{A})$ is proper, every aligned edge of G is 0-crossed. Thus, no edge of G contains I in its interior. Moreover, since $(G, \mathcal{A})$ is a proper triangulation, the outer face of G does not contain intersections of $\mathcal{A}$. Hence, there is a triangular face f of G that is not the outer face and that contains I. Thus, f either has a 2-anchored edge, a 1-anchored l_1-crossed edge, $l_1 \geq 1$, or an l_0-crossed edge, $l_0 \geq 2$, on its boundary. This contradicts that $(G, \mathcal{A})$ has alignment complexity $(1, 0, \perp)$.

Claim 2. Every cell contains at least one free vertex.

Proof of the Claim. Let C be a cell of $\mathcal{A}$. Assume that the boundary of C is neither covered by 1-aligned edges nor crossed by an edge. Since $(G, \mathcal{A})$ is proper, there is a face f of G that entirely contains C in its interior. Further, G is triangulated and therefore, f is a triangle. But every triangle that contains a cell C in its interior either has a 2-anchored edge, a 1-anchored l_1-crossed edge, $l_1 \geq 1$, or an l_0-crossed edge, $l_0 \geq 2$, on its boundary. The alignment complexity of $(G, \mathcal{A})$ excludes these types of edges, thus, there is either a 1-crossed edge with an interior intersection with the boundary of C, or C is covered by 1-anchored aligned edges.

If there is an edge e with an interior intersection with the boundary of C, one endpoint of e lies in the interior of C. Thus, in the following we can assume that



(a) (b) (c)

Figure 10.16: Illustrations for the proof of Lemma 10.17.

no such edges exist. Therefore, the boundary of C is covered by 1-anchored aligned edges. There are two possibilities to triangulate the interior of the cell, either by edges routed through the interior of C with endpoints on the boundary of C or with interior vertices. The former is not possible, since such a non-aligned edge would either be 2-anchored or have both of its endpoints on the same pseudoline. Since $(G, \mathcal{A})$ is an aligned graph of alignment complexity $(1, 0, \perp)$, it does not contain such edges. Thus, every proper aligned triangulation of the graph induced by edges on the boundary of C contains a vertex in the interior of C. ◁

Claim 3. Every cell contains at most one free vertex.

Proof of the Claim. The following proof is similar to Claim 2 in the proof of Lemma 10.15. Let C be a cell and assume for the sake of a contradiction that C contains more than one vertex in its interior; see Figure 10.16a. These vertices are connected by a set of edges to adjacent cells. If C contains a vertex v or an edge e on its boundary, we reroute the corresponding pseudolines close to v and e, respectively, such that v and e are now outside of C; refer to Figure 10.16b. Let C' be the resulting cell, it represents a cut in the graph with two components A and B, where C' contains B in its interior. It is not difficult to see that the modified pseudolines are still pseudolines with respect to G. Since $(G, \mathcal{A})$ neither contains 2-anchored edges, nor 1-anchored l_1-crossed edges, $l_1 \geq 1$, nor l_0-crossed edges, $l_0 \geq 2$, every edge of $(G, \mathcal{A}')$ intersects the boundary of C' at most once. Further, G is a triangulation and therefore, B is connected and since it contains at least two vertices it also contains at least one free edge, contradicting our initial assumption. ◁

Claim 4. Every flexible aligned vertex is incident to two 1-anchored aligned edges.

Proof of the Claim. Let v be a flexible aligned vertex that lies on a pseudosegment $\mathcal{S}$ of $\mathcal{A}$; refer to Figure 10.16c. Since $k \geq 2$, $\mathcal{S}$ is either incident to one or two intersection vertices. Let u be an intersection vertex incident to $\mathcal{S}$ and let $\mathcal{S}$ be on the boundary of the cells C_1, C_2. First, we will show that u is adjacent to a vertex x in the interior of C_1 and a vertex y in the interior of C_2, respectively. Depending on whether $\mathcal{S}$ is incident to one or two intersection vertices, the edge ux helps to find either a separating triangle or a 4-cycle that each contains v in its interior.

We initially show that the graph contains the edge ux. Since G is triangulated there is a fan of triangles around u. Further, all edges in $(G, \mathcal{A})$ are at most 1-crossed, hence we find a vertex x' in the interior of C_1. Due to Claim 3 and Claim 4 the vertex in the interior of C_1 is unique. Thus, we have that x' is equal to x and therefore G contains the edge ux. Correspondingly, we find a vertex y in the interior of C_2 adjacent to u.

Consider the case where $\mathcal{S}$ contains only a single intersection vertex, i.e, $\mathcal{S}$ intersects the outer face of G. Since $(G, \mathcal{A})$ is proper (edges on the outer face are 1-crossed), G contains the edge xy. Thus, we find a triangle with the vertices x, y and u that contains v in its interior. This contradicts the assumption that G does not have a separating triangle. Therefore, if $\mathcal{S}$ is incident to a single intersection, there is no flexible aligned vertex that lies in the interior of $\mathcal{S}$.

Now consider the case where $\mathcal{S}$ is incident to two intersection vertices u and w. As shown before, the vertices u, w are each adjacent to the free vertices x and y. Therefore, vertices u, w, x, y build a 4-cycle containing v in its interior. Since G does not contain a separating triangle, it cannot contain the edge xy. Moreover, v is the only vertex in the interior of $\mathcal{S}$, as otherwise, we would find a free aligned edge. Finally, since $(G, \mathcal{A})$ is an aligned triangulation, the vertex v is connected to all four vertices and thus v is incident to two 1-anchored aligned edges. ◁

Claim 1 proves that $(G, \mathcal{A})$ has Property (i). Claim 2 and Claim 3 together prove that Property (ii) is satisfied. Since $(G, \mathcal{A})$ is an aligned triangulation, Property (iii) immediately follows from Property (ii) and Claim 4. □

Lemma 10.18. *Let $(G, \mathcal{A})$ be a proper k-aligned triangulation of alignment complexity $(1, 0, \perp)$ that does neither contain a free edge, nor a 0-anchored aligned edge, nor a separating triangle. Let A be a line arrangement homeomorphic to the pseudoline arrangement $\mathcal{A}$. Then $(G, \mathcal{A})$ has an aligned drawing (Γ, A).*

Proof. We obtain a drawing (Γ, A) by placing every free vertex in its cell, every aligned vertex on its pseudosegment and every intersection vertex on its intersection. According to Lemma 10.17 every cell and every intersection contains exactly one vertex and each pseudosegment is either crossed by an edge or it is covered by two aligned edges. Observe that the union of two adjacent cells of the arrangement A is convex. Thus, this drawing of G has an homeomorphic embedding to $(G, \mathcal{A})$ and every edge intersects in (Γ, A) the line $L \in A$ corresponding to the pseudoline $C \in \mathcal{A}$ in $(G, \mathcal{A})$ □

We prove the following theorem along the same lines as Theorem 10.16.

Theorem 10.19. *Every k-aligned graph $(G, \mathcal{A})$ of alignment complexity $(1, 0, \perp)$ with a stretchable pseudoline arrangement $\mathcal{A}$ has an aligned drawing.*

Proof. Let $(G, \mathcal{A})$ be an arbitrary aligned graph, such that $\mathcal{A}$ is a stretchable pseudoline arrangement, let us denote by A the corresponding line arrangement. By Lemma 10.12, we obtain a proper k-aligned triangulation $(G_T, \mathcal{A})$ that contains a rigid subdivision of G as a subgraph. Assume that $(G_T, \mathcal{A})$ has an aligned drawing (Γ_T, A). Let (Γ', A) be the drawing obtained from (Γ_T, A) by removing all subdivision vertices v and merging the two edges incident to v at the common endpoint. Recall that a subdivision vertex in a rigid subdivision of $(G, \mathcal{A})$ lies on an intersection in $\mathcal{A}$. Hence the drawing (Γ', A) is a straight-line aligned drawing and contains an aligned drawing (Γ, A) of $(G, \mathcal{A})$.

We now show that $(G_T, \mathcal{A})$ indeed has an aligned drawing. We prove this by induction on the size of the instance $(G_T, \mathcal{A})$. If $(G_T, \mathcal{A})$ neither contains a free edge, nor a 0-anchored aligned edge, nor a separating triangle, then, by Lemma 10.18 there is an aligned drawing (Γ_T, A).

If G contains a separating triangle T, let G_{in} and G_{out} be the respective split components with $G_{\text{in}} \cap G_{\text{out}} = T$. Since the alignment complexity of $(G, \mathcal{A})$ is $(1, 0, \perp)$, triangle T is intersected by at most one pseudoline C. It follows that $(G_{\text{out}}, \mathcal{A})$ is a k-aligned triangulation and that (G_{in}, C) is a 1-aligned triangulation. By the induction hypothesis there exists an aligned drawing (Γ_{out}, A) of $(G_{\text{out}}, \mathcal{A})$. Let $\Gamma_{\text{out}}[T]$ be the drawing of T in Γ_{out}. By Theorem 10.16, we obtain an aligned drawing (Γ_{in}, L) with T drawn as $\Gamma_{\text{out}}[T]$. Moreover, since the drawing of T is fixed and is intersected only by line L, (Γ_{in}, A) is an aligned drawing. Thus, according to Lemma 10.13, there exists an aligned drawing of $(G, \mathcal{A})$.

If G_T does not contain separating triangles but contains either a free edge or a 0-anchored aligned edge e, let G_T/e be the graph after the contraction of e. Observe that, since $(G_T, \mathcal{A})$ is proper, every edge on the outer face is 1-crossed, and therefore every chord is ℓ-crossed, $\ell \geq 1$. Thus, e is an interior edge of $(G_T, \mathcal{A})$ and is not a chord. Therefore, by Lemma 10.14, $(G_T/e, \mathcal{A})$ is a proper aligned triangulation. By induction hypothesis, there exists an aligned drawing of $(G_T/e, \mathcal{A})$, and thus, by the same lemma, there exists an aligned drawing of $(G_T, \mathcal{A})$. □

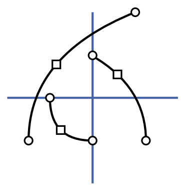


Figure 10.17: Placement of a subdivision vertex to obtain a 2-aligned graph of alignment complexity $(1, 0, \perp)$.

Theorem 10.20. *Every 2-aligned graph has an aligned drawing with at most one bend per edge.*

Proof. We subdivide 2-crossed, 2-anchored or 1-crossed 1-anchored edges as depicted in Figure 10.17. Thus, we obtain a 2-aligned graph $(G', \mathcal{A})$ of alignment complexity $(1, 0, \perp)$. Applying Theorem 10.19 to $(G', \mathcal{A})$ yields a one bend drawing of $(G, \mathcal{A})$. □

10.4.4 Aligned Drawings of Counterclockwise Aligned Graphs

In this section, we consider aligned drawings of 2-aligned graphs $(G, \mathcal{A})$, i.e., $\mathcal{A} = \{\mathcal{X}, \mathcal{Y}\}$ where $\mathcal{X}$ and $\mathcal{Y}$ are two intersecting pseudolines with respect to G. For convenience, we denote a 2-aligned graph as $(G, \mathcal{XY})$. Recall that, the aligned graph in Figure 10.18a does not have an aligned drawing. The crux is that the source of the red edges are free and the source of green edges are aligned. In the following we

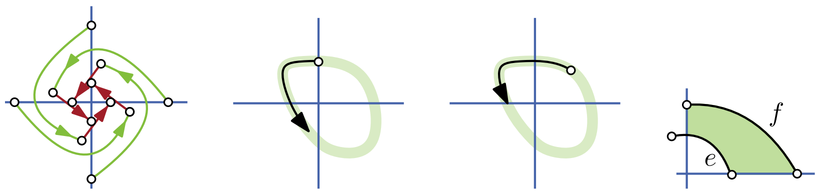


Figure 10.18: (a) This 2-aligned graph does not have an aligned drawing. (b,c) The green curve indicates the Jordan curve that completes the black edge. The edge in (b) is an edge of a ccw-aligned graph. The edge depicted in (c) is forbidden in ccw-aligned graphs. (d) A comb of edges e, f.

introduce so-called *counterclockwise aligned graphs* and show that they have aligned drawings.

We orient each non-aligned edge uv of an aligned graph $(G, \mathcal{X}\mathcal{Y})$ such that it can be extended to a Jordan curve, i.e., a closed simple curve, C_{uv} with the property that it intersects each pseudoline exactly twice and has the origin to its left. A *counterclockwise aligned (ccw-aligned)* graph is a 2-aligned graph of alignment complexity $(1, 1, 0)$ whose orientation does not contain 1-anchored 1-crossed edges with a free source vertex.

We prove that every ccw-aligned graph has an aligned drawing. To prove this statement we follow the same proof strategy as before with minor modifications. In particular, we have to ensure that there is a proper ccw-aligned triangulation. Then in Lemma 10.23 we show that for each aligned graph $(G, \mathcal{X}\mathcal{Y})$ there is a *reduced aligned graph* $(G_R, \mathcal{X}\mathcal{Y})$ (i.e., it does neither contain (i) separating triangles, (ii) free edges, and (iii) aligned edges that are not incident to the intersection of $\mathcal{X}$ and $\mathcal{Y}$) with the property that $(G, \mathcal{X}\mathcal{Y})$ has an aligned drawing if $(G_R, \mathcal{X}\mathcal{Y})$ has an aligned drawing. In contrast to aligned graphs of alignment complexity $(1, 0, \perp)$ the size of $(G_R, \mathcal{X}\mathcal{Y})$ is not bounded by a constant. Thus, the main contribution of this section is Lemma 10.26 that states that each reduced instance has an aligned drawing.

We first introduce some further notations. We refer to the intersection of $\mathcal{X}$ and $\mathcal{Y}$ as the *origin* $\mathcal{O}$. The curves $\mathcal{X}, \mathcal{Y}$ divide the plane into four quadrants $\mathcal{Q}_1, \ldots, \mathcal{Q}_4$ in counterclockwise order. These quadrants naturally correspond to the regions $Q_1, \ldots, Q_4$ bounded by the lines X and Y. Additionally to the prior definition of proper k-aligned graphs (Section 10.4.1), we now require that there is a degree-4 vertex o on the origin that is incident to four aligned edges. Moreover, we require that the outer face is bounded by 2-anchored edges instead of 1-crossed edges. Thus, in the following a 2-aligned graph $(G, \mathcal{X}\mathcal{Y})$ is a *proper 2-aligned triangulation* if each inner face is a triangle, the boundary of the outer face is a 4-cycle of 2-anchored edges, the outer face does not contain the origin and there is a degree-4 vertex o on the origin incident to four aligned edges. We refer to a reduced proper ccw-aligned triangulation as a

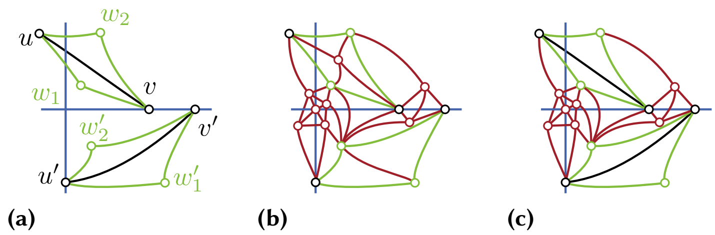


Figure 10.19: (a) The (black) separating edges are isolated by the green edges. (b) The black edges are removed and the red edges are obtained by the triangulation. (c) Final graph, after removing edges in the interior of a quadrangle u, w_1, v, w_2 and reinserting the black edges.

reduced aligned triangulation. We refer to 1-anchored 1-crossed and 2-anchored edges as *separating*. The region within a quadrant that is bounded by two separating edges e and f is an *edge region*; see Figure 10.18d. An inclusion-minimal edge region is a *comb*.

Lemma 10.21. *Let $(G, \mathcal{X}\mathcal{Y})$ be a ccw-aligned graph. Then there is a ccw-aligned triangulation $(G', \mathcal{X}\mathcal{Y})$ that contains $(G, \mathcal{X}\mathcal{Y})$ as a subgraph. Moreover, the outer face of $(G', \mathcal{X}\mathcal{Y})$ is bounded by 4-cycle C of 2-anchored edges and the outer face does not contain the origin in its interior.*

Proof. Let $(G_2, \mathcal{X}\mathcal{Y})$ be the graph that is constructed from $(G, \mathcal{X}\mathcal{Y})$ as follows. First, add a 4-cycle C of 2-anchored edges in the outer face such that the new outer face does not contain the origin. For each separating edge uv of G add two vertices w_1, w_2 and the edges uw_1, w_1v and uw_2, w_2v. Route and direct the edges according to Figure 10.19a. Finally, remove the edge uv. Eventually, we arrive at an aligned graph of alignment complexity $(1, 0, \perp)$. With the application of Lemma 10.9 and Lemma 10.19 we obtain a triangulated aligned graph $(G_3, \mathcal{X}\mathcal{Y})$ of alignment complexity $(1, 0, \perp)$. We remove edges in the interior of each quadrangle u, w_1, v, w_2 and reinserted the original edge uv. Finally, we remove all edges and vertices in the region bounded by C that does not contain the origin. This yields the desired aligned graph $(G', \mathcal{X}\mathcal{Y})$. ◻

Since no free edge of an ccw-aligned graph is incident to a triangle that contains the intersection in its interior, the following lemma can be proven along the same lines as Lemma 10.14.

Lemma 10.22. *Let $(G, \mathcal{X}\mathcal{Y})$ be a ccw-aligned graph and let e be an interior free edge or an aligned edge that is neither an edge of a separating nor a chord and does not contains the origin, then $(G/e, \mathcal{X}\mathcal{Y})$ is a ccw-aligned graph and $(G, \mathcal{X}\mathcal{Y})$ has an aligned drawing if $(G/e, \mathcal{X}\mathcal{Y})$ has an aligned drawing.*

With the tools introduced in Section 10.4.1 we can now prove the following lemma.

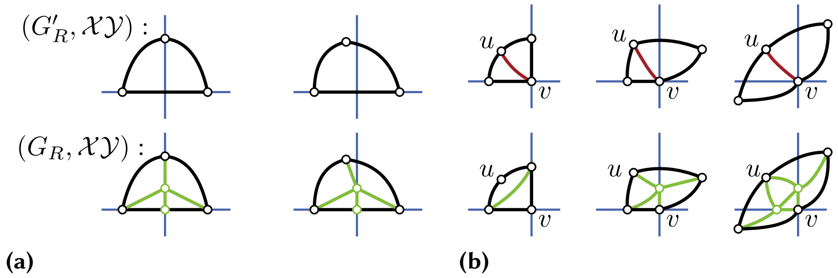


Figure 10.20: Red edges are removed from $(G_T, \mathcal{XY})$ and green added to $(G_P, \mathcal{XY})$

Lemma 10.23. *For every ccw-aligned graph $(G, \mathcal{XY})$ there is a reduced aligned triangulation $(G_R, \mathcal{XY})$ such that $(G, \mathcal{XY})$ has an aligned drawing if $(G_R, \mathcal{XY})$ has an aligned drawing.*

Proof. By Lemma 10.21 there is a aligned triangulation $(G_T, \mathcal{XY})$ of $(G, \mathcal{XY})$ with the outer face bounded by 4-cycle of 2-anchored edges. Moreover, an aligned drawing of $(G_T, \mathcal{XY})$ contains an aligned drawing of $(G, \mathcal{XY})$.

By the application Lemma 10.13 and Lemma 10.22, we obtain a reduced aligned triangulation $(G'_R, \mathcal{XY})$ from $(G_T, \mathcal{XY})$ by either splitting $(G_T, \mathcal{XY})$ into two aligned graphs at a separating triangle T, or by contracting free or aligned edges that are not incident to o. Note that, again by the same lemmas, we have that that $(G_T, \mathcal{XY})$ has an aligned drawing if $(G'_R, \mathcal{XY})$ has an aligned drawing

In order to obtain a proper aligned triangulation $(G_R, \mathcal{XY})$ from $(G'_R, \mathcal{XY})$ we perform the reduction depicted in Figure 10.20. If there is an aligned edge that contain the origin in its interior, we place a subdivision vertex on this edge and inserted edges as depicted in Figure 10.20a. Note that in this case an aligned drawing of $(G_R, \mathcal{XY})$ contains an aligned drawing of $(G'_R, \mathcal{XY})$.

Consider the case that there is a vertex v on the origin that is incident to a free vertex u. We obtain a new aligned graph $(G_R, \mathcal{XY})$ by exhaustively applying the reductions depicted in Figure 10.20b. Since the black polygon (compare Figure 10.20b) in an aligned drawing of $(G_R, \mathcal{XY})$ is star-shaped and its kernel contains the vertex v, $(G'_R, \mathcal{XY})$ has an aligned drawing if $(G'_R, \mathcal{XY})$ has an aligned drawing. □

The following lemma describes the structure of reduced triangulations.

Lemma 10.24. *Let $(G_R, \mathcal{XY})$ be a reduced aligned triangulation and let o be the vertex on the origin. Then in $(G_R - o, \mathcal{XY})$ the pseudolines $\mathcal{X}$ and $\mathcal{Y}$ alternately intersect vertices and edges, and each comb contains at most one vertex.*

Proof. Assume that there are two consecutive aligned vertices u and v. Since G is triangulated and u and v are consecutive, G contains the edge uv. This contradicts the assumption that $(G, \mathcal{XY})$ does not contain aligned edges.

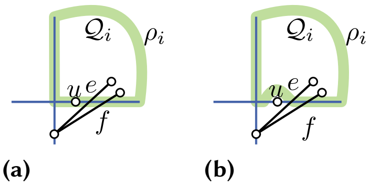


Figure 10.21: The curve ρ_i (a) and its modification in (b).

The following modification helps us to prove that there are no two consecutive edges along a pseudoline and that no comb contains two free vertices.

Let ρ_i be the parts of $\mathcal{X}$ and $\mathcal{Y}$ that are on the boundary of the quadrant $\mathcal{Q}_i$, see Figure 10.21. We modify ρ_i as follows. We first, join the endpoints of ρ_i in the infinity such that it becomes a simple closed curve. Let u be a vertex that lies on ρ_i. We reroute ρ_i such that u now lies outside of ρ_i. Since G is triangulated and ρ_i only intersects edges, ρ_i corresponds to a cycle in $G^\star$ and therefore to a cut C_i in G. Note that each edge of a connected component in $G - C_i$ is a free edge.

Now assume that there are two distinct edges e, f that consecutively cross a pseudoline $\mathcal{L} \in \mathcal{XY}$. By the premises of the lemma there is a vertex that lies on the origin $\mathcal{O}$. Hence both e and f cross $\mathcal{L}$ on the same side with respect to $\mathcal{O}$. Since e and f are distinct and $(G, \mathcal{XY})$ is ccw-aligned, there is a quadrant $\mathcal{Q}_j$ such that $\mathcal{Q}_j$ contains two distinct vertices u and w incident to e and f, respectively. Since G is triangulated and e and f are consecutive along $\mathcal{L}$, u and w are vertices in the same connected component of $G - C_j$. Therefore, $(G, \mathcal{XY})$ contains a free edge. A contradiction.

Consider a comb C in a quadrant $\mathcal{Q}_i$ that contains two distinct vertices u and v in its interior. Since G is triangulated and C is inclusion-minimal (it does not contain another edge-region), u and v belong to the same connected component of $G - C_i$. Therefore $(G, \mathcal{XY})$ contains a free edge. □

We call a comb *closed* if its two separating edges have the same source vertex.

Lemma 10.25. *For every reduced aligned triangulation $(G_R, \mathcal{XY})$ there is a reduced aligned triangulation $(G''_R, \mathcal{XY})$ where no closed comb contains a vertex such that $(G_R, \mathcal{XY})$ has an aligned drawing if $(G''_R, \mathcal{XY})$ has an aligned drawing.*

Proof. By Lemma 10.24 we know that each comb contains at most one vertex. We apply induction over the number of closed combs that contain a vertex. Let v be a free vertex in a closed comb with separating edges uw_1, uw_2. Then we obtain an aligned graph $(G'_R, \mathcal{XY})$ by contracting edge uv in the embedding. Since $(G_R, \mathcal{XY})$ is reduced ccw-aligned, all edges outgoing from the free vertex v are 1-anchored 0-crossed or 0-anchored 1-crossed. In $(G'_R, \mathcal{XY})$ they are now 2-anchored 0-crossed or 1-anchored 1-crossed with free target vertex. Since there is no other vertex in the comb and

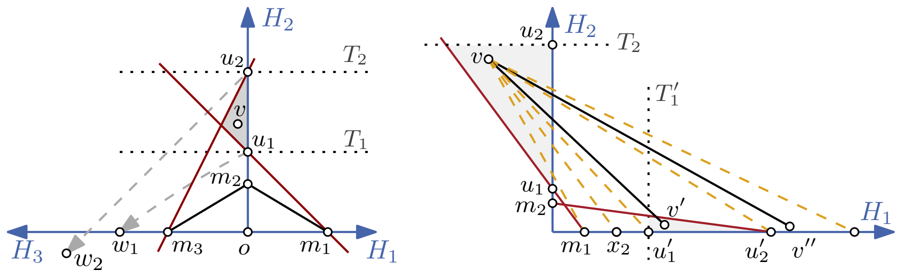


Figure 10.22: (a) Placement of a free vertex v in quadrant Q_2. It may be placed within the gray triangle. (b) Example for the observations with $u_1' = x_3$ and $u_2' = x_4$.

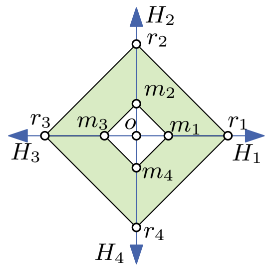


Figure 10.23: The vertex o and the half-lines H_i and the vertices m_i, r_i for $i = 1, \ldots, 4$. All remaining edges and vertices lie in the green area.

the comb is closed, v only has uv as incoming edge which is contracted. Therefore $(G_R', \mathcal{XY})$ is ccw-aligned. Assume that $(G_R', \mathcal{XY})$ has an aligned drawing. Since v is a free vertex, we obtain an aligned drawing of $(G, \mathcal{XY})$ by placing v close to u within in its closed comb. By Lemma 10.23 we obtain a reduced aligned triangulation $(G_R'', \mathcal{XY})$ from $(G', \mathcal{XY})$ such that $(G_R', \mathcal{XY})$ has an aligned drawing if $(G_R'', \mathcal{XY})$ has an aligned drawing. In the construction the number of closed combs that contain a vertex is not increased. This concludes the induction. □

We can now show that each reduced instance has an aligned drawing which is the core of our contribution.

Lemma 10.26. *Every reduced aligned triangulation has an aligned drawing.*

Proof. By Lemma 10.25 we can assume that in our triangulation $(G, \mathcal{XY})$ the closed combs contain no vertices. By Lemma 10.24 we know that each comb contains at most one vertex and no vertex if it is closed. The main problem is to draw the 1-crossed edges. For those, we place each free vertex v close to the right boundary of its comb.

This allows to draw the incoming edges. Since $(G, \mathcal{XY})$ is ccw-aligned, the target of each 1-crossed edge vu is free and allows to draw vu.

We construct the aligned drawing (Γ, XY) as follows. Let o be the vertex on the origin. We call the sources of separating edges *corners*. First place o and all corners on XY in the order induced from $\mathcal{XY}$. For $i = 1, \ldots, 4$, let $\mathcal{H}_i$ be the half-pseudoline that is the right boundary of quadrant Q_i. Let m_i denote the vertex on $\mathcal{H}_i$ that is adjacent to o and let r_i denote the vertex incident to the outer face on $\mathcal{H}_i$. Note that m_i, r_i are corners. We write $u <_i v$ if u lies between o and v on $\mathcal{H}_i$ where u, v may be vertices and intersections of edges with $\mathcal{H}_i$. Note that $<_i$ is a linear order. Define H_i correspondingly for XY; see Figure 10.23. The indices for m_i, Q_i, etc. are considered mod 4. In the following, we denote by $\overline{uv}$ the line through two distinct points u, v. Now consider a free vertex v in some quadrant Q_i; see Figure 10.22a. It lies in a comb that is bounded by two separating edges u_1w_1, u_2w_2 with $u_1 <_i u_2$ on $\mathcal{H}_i$. Note that we have $u_1 \neq u_2$ since the comb contains v and is thus not closed. We place v within the triangle bounded by $\overline{m_{i+1}u_2}$, $\overline{m_{i-1}u_1}$ and H_i. Note that v lies in Q_i and between the two lines T_1, T_2 through u_1, u_2 that are orthogonal to H_i. We will show that the intersections of 1-crossed edges with H_i and the corners on H_i respect the order $<_i$. Finally, we place for $i = 1, \ldots, 4$ the vertices on $\mathcal{H}_i$ that are neither o nor a corner arbitrarily on H_i respecting the order $<_i$. This finishes the construction (edges are placed accordingly).

We next show that the vertices and edges of G appear along X and $\mathcal{X}$ (respectively Y and $\mathcal{Y}$) in the same order. Consider the free vertex v and the separating edges u_1w_1, u_2w_2 as defined above. Let $m_{i-1} = x_1 <_{i-1} \cdots <_{i-1} x_k = r_{i-1}$ denote the corners on H_{i-1}. The following three observations imply that all 1-crossed edges with target v cross H_i in the correct order between u_1 and u_2; refer to Figure 10.22b.

1. $\overline{m_{i-1}v}$ and $\overline{r_{i-1}v}$ cross H_i between u_1 and u_2.
2. $\overline{x_1v}, \ldots, \overline{x_kv}$ intersect H_i in the same order as $x_1, \ldots, x_k$ lie on $\mathcal{H}_{i-1}$.
3. Let v' be a free vertex in Q_{i-1}. Let $u_1'w_1'$, $u_2'w_2'$ be the separating edges of the comb containing v'. Then $v'v$ crosses H_i between $\overline{u_1'v} \cap H_i$ and $\overline{u_2'v} \cap H_i$.

For Observation 1, note that v lies in the triangle bounded by H_i, $\overline{m_{i-1}u_1}$ and T_2. For Observation 2, note that $\overline{x_1v}, \ldots, \overline{x_kv}$ cross pairwise in v and thus not in quadrant Q_{i-1}. These two observations imply that $\overline{x_1v}, \ldots, \overline{x_kv}$ cross H_{i-1} between u_1 and u_2. For Observation 3 note now that v' lies in the triangle bounded by H_{i-1}, $\overline{u_2'm_i}$ and the line T_1' through u_1' that is orthogonal to H_{i-1}.

We now show that all 1-crossed edges with target v cross H_i in the correct order between u_1 and u_2. By Observations 2, 3 the 1-crossed edges with target v cross H_i between $\overline{m_{i-1}v} \cap H_i$ and $\overline{r_{i-1}v} \cap H_i$. With Observation 1, they cross H_i between u_1 and u_2. By Observation 2, we know that the 1-anchored 1-crossed edges with target

v cross H_i in the correct order. By Observations 2, 3, we obtain that each pair of a 0-anchored 1-crossed and a 1-anchored 1-crossed edge cross H_i in the correct order. Since the sources of 0-anchored 1-crossed edges with target v lie in different combs, they lie pairwise on different sides of some edge $x_j v$ by Observation 3. Observation 2 then yields their correct ordering.

Since the corners on H_i respect $<_i$ and all 1-crossed edges have free target vertices (as the triangulation is ccw-aligned), this implies that the intersections of 1-crossed edges with H_i and the corners on H_i respect the order $<_i$. By construction, we placed the vertices on $\mathcal{H}_i$ that are not corners such that they also respect order $<_i$. Thus, X, Y intersect the vertices and edges in the same order as $\mathcal{X}$, $\mathcal{Y}$.

We next show that our embedding is planar by showing that there is no location where edges cross. Since the order of intersections with XY is correct, there are no crossings on $X \cup Y$. This leaves us with the quadrants. Since the separating edges of $\mathcal{Q}_i$ appear in the same order on $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$, they also appear in the same order on H_i and H_{i+1}. Thus, separating edges of the same quadrant do not cross each other. We further obtain the same combs for (Γ, XY). Consider again a free vertex v in Q_i and the corresponding separating edges u_1w_1, u_2w_2; see Figure 10.22a. Since v lies in the triangle bounded by H_i, T_1 and $\overline{m_{i+1}u_2}$, it also lies in the comb bounded by u_1w_1, u_2w_2. Hence, every free vertex lies in the correct comb. Let e be an edge incident to v. Then its other end vertex does not lie within the comb of v. It must therefore intersect $\mathcal{H}_i$ between u_1 and u_2 if it is incoming, and it must intersect $\mathcal{H}_{i+1}$ between $u_1w_1 \cap \mathcal{H}_{i+1}$ and $u_2w_2 \cap \mathcal{H}_{i+1}$ if it is outgoing. Since we have the same order on H_i and H_{i+1} respectively, edge e crosses neither u_1w_1 nor u_2w_2 and thus not the interior of any other comb in Q_i. This means that 1. There are no crossings on separating edges in the corresponding quadrants. And that 2. Only edges incident to the free vertex v in a comb intersect the interior of that comb. Those edges are all adjacent in v and do not cross. We obtain that there are no crossings on $X \cup Y$, no crossings on separating edges in the corresponding quadrants and no crossings within combs. Hence, our embedding is planar.

Since there are no free edges and the order of intersections with XY is fixed, the order of incident edges around a free vertex is also fixed. For a vertex u on XY we note that each adjacent free vertex is in another comb and therefore the order of incident edges around u is also fixed. Therefore, our embedding Γ induces the same combinatorial embedding as the embedding of G. □

From Lemma 10.23 and Lemma 10.26 we directly obtain our main theorem.

Theorem 10.27. *Every ccw-aligned graph* $(G, \mathcal{X}\mathcal{Y})$ *has an aligned drawing.*

10.5 Conclusion

In this chapter, we showed that if $\mathcal{A}$ is stretchable, then every k-aligned graph $(G, \mathcal{A})$ of alignment complexity $(1, 0, \bot)$ has a straight-line aligned drawing. As an intermediate result, we showed that a 1-aligned graph $(G, \mathcal{R})$ has an aligned drawing with a fixed convex drawing of the outer face. We showed that the less restricted version of this problem, where we are only given a set of vertices to be aligned, is $\mathcal{NP}$-hard but fixed-parameter tractable.

There is a 2-aligned graph of alignment complexity $(1, 1, 0)$ that does not have an aligned drawing. Every 2-aligned graph of alignment complexity $(1, 0, 0)$, and the more general ccw-aligned graphs, have aligned drawings. Table 10.1 summarizes these results. It is open whether this result can be extended to k-aligned graphs of alignment complexity $(1, 0, 0)$.

Overall, we state the following open questions.

1) What are all the combinations of line numbers k and alignment complexities C such that for every k-aligned graph $(G, \mathcal{A})$ of alignment complexity C there exists a straight-line aligned drawing provided $\mathcal{A}$ is stretchable?
2) Given a k-aligned graph $(G, \mathcal{A})$ and a line arrangement A homeomorphic to $\mathcal{A}$, what is the computational complexity of deciding whether $(G, \mathcal{A})$ admits a straight-line aligned drawing (Γ, A)?

11 Conclusion and Open Problems

In this thesis, we studied geometric graph drawing algorithms from a theoretical and practical perspective. The algorithms in Part I are concerned with crossings in geometric drawings, in particular with the number of crossings. In Part II, we studied whether topological planar embeddings can be stretched to geometric drawings while satisfying prescribed constraints.

In Part I, we introduced and evaluated algorithms that compute drawings with a small number of crossings in small and large graphs; see Chapter 4 and Chapter 5. In Chapter 6, we traded the requirement that the edges have to be straight-line segments for a smaller number of crossings and studied algorithms that stretch a topological drawing with few crossings to a drawing where the edges are as straight as possible. For all these approaches, we were able to show that the quality of the drawings is considerably better than the quality of drawings obtained by energy-based approaches. It is unclear how close the computed drawings are to an optimal solution. In case of the number of crossings in geometric drawings, a study of the relationship between the values $\mathrm{cr}(\Gamma)$ and $\mathrm{cr}(\Gamma^\star)$ would be of interest, where Γ is a drawing computed by a heuristic and $\Gamma^\star$ is a drawing with a minimum number of crossings. This question can be approached from two directions. One is to compute an optimal solution for each graph in the benchmark set. The other direction is to prove bounds for $\mathrm{cr}(\Gamma^\star)$ for special graph classes and then to apply the heuristic to these graphs. The facts that the geometric crossing minimization problem is $\exists\mathbb{R}$-complete [Bie91], the 700 references in Vrt'o's [Vrt14] online-bibliography to papers studying crossings and that the geometric crossing number of the complete graph on n vertices is only known for $n \leq 27$ and $n = 30$ [Aic19], indicate that these sort of problems are indeed challenging. Nevertheless, we think the following questions are worth considering. Given a planar graph $G = (V, E)$ and a set X of additional edges, i.e., $E \cap X = \emptyset$, is it fixed-parameter tractable in $|X|$ to compute the geometric crossing number of $G + X$? Or more specific, is the problem tractable if G is a planar triangulation (i.e., the class Triangulation+X in benchmark set used in Part I), or when X contains only a single edge? Can we prove bounds for the geometric crossing number of these instances?

A popular recent research trend is to consider beyond planar graphs, e.g., *k-planar graphs* that have a drawing with at most k crossings per edge. These problems are often studied from a topological perspective. For 1-planar graphs there is a polynomial-time algorithm that either reports that an embedded 1-planar graph does not have 1-planar geometric drawing or it returns a 1-planar geometric drawing [Hon+12]. Thus, for

1-planar graphs there is a baseline for our crossing-minimization heuristics, if we do not consider the number of crossings in geometric drawings but the maximum number of crossings per edge in a geometric drawing, i.e., the *local crossing number* of a drawing. Hence, with the aim to provide further indications that the vertex-movement approach is indeed a powerful graph drawing heuristic, we ask the following question: Can the vertex-movement approach be successfully applied to compute geometric drawings with a small local crossing number?

The first problem in Part II considers the previously stated problem of computing crossing-minimal geometric drawing of a graph $G + X$ from a restricted perspective. In the setting in Chapter 8, we are given a topologically embedded graph G and a single additional edge e, i.e., $X = \{e\}$. The problem asks for a geometric drawing Γ that has the same combinatorial embedding and outer face as G and such that the edge e can be embedded as straight-line segment in Γ with a minimum number of crossings. We solved this problem for special cases. The general computational complexity of the problem remains open, i.e., is the problem $\mathcal{NP}$-complete or in $\mathcal{P}$? Moreover, the question whether a crossing-minimal topological drawing $\mathcal{E} + e$ of $G + e$, where the drawing $\mathcal{E}$ of G is planar, has a geometric drawing $\Gamma + e$ with the same number of crossings, and $\mathcal{E}$ and Γ have the same combinatorial embedding, is closely related to an edge-disjoint-path problem: Are there two non-crossing edge-disjoint st-paths p and p' in a planar graph such that the length of p is minimized? For general graphs this problem is $\mathcal{NP}$-complete [Eil98]. For planar graphs we solved this problem if all shortest paths from s to t have a specific structure. The computational complexity of this problem for general planar graphs is an intriguing unsolved problem.

The paths p and p' together correspond to a pseudoline $\mathcal{L}$ with respect to a planar graph G. In Chapter 10, we proved that every 1-aligned graph $(G, \mathcal{L})$ has an aligned drawing with the prescribed convex drawing of the outer face. A natural extension of the geometric edge-insertion problem is to ask whether a planar graph G has a geometric drawing such that two distinct edges e_1 and e_2 can be inserted into the drawing with a minimal number of crossings. In Chapter 10 we showed that not every 2-aligned graph has an aligned drawing. Therefore, in case that e_1 and e_2 share an endpoint, it is not sufficient to ask for a 2-aligned graph $(G, \mathcal{XY})$ such that two rays of $\mathcal{XY}$ have a minimal number of crossings. Thus, we ask for a characterization of topologically embedded graphs $G + \{e_1, e_2\}$ that are stretchable.

We introduced the alignment complexity of an aligned graph $(G, \mathcal{A})$ as a concept to describe the interplay of the edges of G with the pseudoline arrangement $\mathcal{A}$. We proved that every aligned graph $(G, \mathcal{A})$ of alignment complexity $(1, 0, \perp)$ has an aligned drawing. If the number of pseudolines is restricted to two, then every aligned graph of aligned complexity $(1, 0, 0)$ has an aligned drawing. Whether this is true for general aligned graphs of alignment complexity $(1, 0, 0)$ is an open question. Answering this would entirely settle the following question: Given an alignment complexity C, does every graph G of alignment complexity C have an aligned drawing? In case that not every graph of alignment complexity C has an aligned drawing, it is certainly possible that there are graphs of alignment complexity C that have an aligned drawing. Thus, a natural question is to ask for the computational complexity of deciding whether an aligned graph has an aligned drawing. Since the problem is closely related to the stretchability of pseudoline arrangements, it would not be too surprising if the problem turned out to be $\mathcal{NP}$-hard. Therefore, it might be more promising to fix the alignment complexity, i.e., given a fixed alignment complexity C, is there a polynomial-time algorithm that decides whether an aligned graph of alignment complexity C has an aligned drawing?

Bibliography

[AAM18] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. **The Art Gallery Problem is ∃ℝ-complete**. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC'18)*, pages 65–73. ACM Press, 2018. DOI: 10.1145/3188745.3188868.
Cited on page 167.

[Ábr+08] Bernardo M. Ábrego, Silvia Fernández–Merchant, Jesús Leaños, and Gelasio Salazar. **The Maximum Number of Halving Lines and the Rectilinear Crossing Number of K_n for n $\leq$ 27**. *Electronic Notes in Discrete Mathematics* 30, pages 261–266, 2008. DOI: 10.1016/j.endm.2008.01.045.
Cited on page 69.

[ABS12] Evmorfia N. Argyriou, Michael A. Bekos, and Antonios Symvonis. **The Straight-Line RAC Drawing Problem is NP-Hard**. *Journal of Graph Algorithms and Applications* 16:2, pages 569–597, 2012. DOI: 10.7155/jgaa.00274.
Cited on pages 5, 97.

[ABS13] Evmorfia N. Argyriou, Michael A. Bekos, and Antonios Symvonis. **Maximizing the Total Resolution of Graphs** 56:7, pages 887–900, 2013. DOI: 10.1093/comjnl/bxs088.
Cited on pages 98, 99, 104.

[AFK11] Patrizio Angelini, Fabrizio Frati, and Michael Kaufmann. **Straight-Line Rectangular Drawings of Clustered Graphs**. *Discrete & Computational Geometry* 45:1, pages 88–140, 2011. DOI: 10.1007/s00454-010-9302-z.
Cited on page 145.

[ÁFS13] Bernardo M. Ábrego, Silvia Fernández-Merchant, and Gelasio Salazar. **The Rectilinear Crossing Number of K_n : Closing in (or Are We?)** In *Thirty Essays on Geometric Graph Theory*. Ed. by János Pach. Springer New York, 2013, pages 5–18. DOI: 10.1007/978-1-4614-0110-0_2.
Cited on pages 23, 24.

[AFT18] Hugo A. Akitaya, Radoslav Fulek, and Csaba D. Tóth. **Recognizing Weak Embeddings of Graphs**. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*. Ed. by Artur Czumaj, pages 274–292. Society for Industrial and Applied Mathematics, 2018. DOI: 10.1137/1.9781611975031.20.
Cited on pages 143, 145.

[Aic19] Oswin Aichholzer. **On the Rectilinear Crossing Number. (http://www.ist.tugraz.at/staff/aichholzer/research/rp/triangulations/crossing)**. Oct. 2019.
Cited on pages 24, 50, 205.

[Aki+17] Hugo A. Akitaya, Greg Aloupis, Jeff Erickson, and Csaba D. Tóth. **Recognizing Weakly Simple Polygons**. *Discrete & Computational Geometry* 58:4, pages 785–821, 2017. DOI: 10.1007/s00454-017-9918-3.
Cited on page 11.

[Ala+15] Md. Jawaherul Alam, Michael Kaufmann, Stephen G. Kobourov, and Tamara Mchedlidze. **Fitting Planar Graphs on Planar Maps**. *Journal of Graph Algorithms and Applications* 19:1, pages 413–440, 2015. DOI: 10.7155/jgaa.00367.
Cited on pages 143, 145, 146, 149.

[Ang+14] Patrizio Angelini, Giordano Da Lozzo, Marco Di Bartolomeo, Giuseppe Di Battista, Seok-Hee Hong, Maurizio Patrignani, and Vincenzo Roselli. **Anchored Drawings of Planar Graphs**. In *Proceedings of the 22nd International Symposium on Graph Drawing (GD'14)*. Ed. by Christian Duncan and Antonios Symvonis. Volume 8871 of Lecture Notes in Computer Science, pages 404–415. Springer Berlin/Heidelberg, 2014.
Cited on pages 7, 143, 144, 146, 167.

[Ari+12] Karin Arikushi, Radoslav Fulek, Balázs Keszegh, Filip Morić, and Csaba D. Tóth. **Graphs that admit Right Angle Crossing Drawings**. *Computational Geometry: Theory and Applications* 45:4, pages 169–177, 2012. DOI: 10.1016/j.comgeo.2011.11.008.
Cited on page 97.

[Bad+18] David A. Bader, Andrea Kappes, Henning Meyerhenke, Peter Sanders, Christian Schulz, and Dorothea Wagner. **Benchmarking for Graph Clustering and Partitioning**. In *Encyclopedia of Social Network Analysis and Mining, 2nd Edition*. Springer-Verlag, 2018. DOI: 10.1007/978-1-4939-7131-2_23.
Cited on page 61.

[Ban+17] Bahareh Banyassady, Michael Hoffmann, Boris Klemz, Maarten Löffler, and Tillmann Miltzow. **Obedient Plane Drawings for Disk Intersection Graphs**. In *Proceedings of the 15th International Symposium on Algorithms and Data Structures (WADS'17)*. Ed. by Faith Ellen, Antonina Kolokolova, and Jörg-Rüdiger Sack. Volume 10389 of Lecture Notes in Computer Science, pages 73–84. 2017.
Cited on pages 144, 149.

[BD93] Daniel Bienstock and Nathaniel Dean. **Bounds for Rectilinear Crossing Numbers**. *Journal of Graph Theory* 17:3, pages 333–348, 1993. DOI: 10.1002/jgt.3190170308.
Cited on pages 5, 23.

[Bek+18] Michael A. Bekos, Henry Förster, Christian Geckeler, Lukas Holländer, Michael Kaufmann, Amadäus M. Spallek, and Jan Splett. **A Heuristic Approach Towards Drawings of Graphs with High Crossing Resolution**. In *Proceedings of the 26th International Symposium on Graph Drawing (GD'18)*. Volume 11282 of Lecture Notes in Computer Science, pages 271–285, 2018. DOI: 10.1007/978-3-030-04414-5_19.
Cited on page 50.

[Ber00] François Bertault. **A Force-Directed Algorithm that Preserves Edge-Crossing Properties**. *Information Processing Letters* 74:1–2, pages 7–13, 2000. DOI: 10.1016/S0020-0190(00)00042-9.
Cited on pages 32, 70.

[Bie91] Daniel Bienstock. **Some Provably Hard Crossing Number Problems**. *Discrete & Computational Geometry* 6:1, pages 443–459, 1991. DOI: 10.1007/BF02574701.
Cited on pages vii, 2, 4, 24, 49, 205.

[BK12] Mark de Berg and Amirali Khosravi. **Optimal Binary Space Partitions for Segments in the Plane**. *International Journal of Computational Geometry & Applications* 22:3, pages 187–206, 2012. DOI: 10.1142/S0218195912500045.
Cited on pages 149, 164.

[BKM98] Therese C. Biedl, Michael Kaufmann, and Petra Mutzel. **Drawing Planar Partitions II: HH-Drawings**. In *Proceedings of the 24th Workshop on Graph-Theoretic Concepts in Computer Science (WG'98)*. Ed. by Juraj Hromkovič and Ondrej Sýkora, pages 124–136. Springer Berlin/Heidelberg, 1998. DOI: 10.1007/10692760_11.
Cited on page 170.

[BM07] Gunnar Brinkmann and Brendan D. McKay. **Fast Generation of Planar Graphs**. *MATCH-Communications in Mathematical and In Computer Chemistry* 58:2. Ed. by Gunnar Brinkmann and Patrick W. Fowler, pages 323–357, 2007. ISSN: 0340-6253.
Cited on page 32.

[BO79] John L. Bentley and Thomas A. Ottmann. **Algorithms for Reporting and Counting Geometric Intersections**. *IEEE Transactions on Computers* C-28:9, pages 643–647, 1979.
Cited on pages 30, 55, 100.

[BP07] Ulrik Brandes and Christian Pich. **Eigensolver Methods for Progressive Multidimensional Scaling of Large Data**. In *Proceedings of the 14th International Symposium on Graph Drawing (GD'06)*. Ed. by Michael Kaufmann and Dorothea Wagner, pages 42–53. Springer Berlin/Heidelberg, 2007. DOI: 10.1007/978-3-540-70904-6_6.
Cited on page 34.

[BR16] Thomas Bläsius and Ignaz Rutter. **A New Perspective on Clustered Planarity as a Combinatorial Embedding Problem**. *Theoretical Computer Science* 609:2, pages 306–315, 2016. DOI: 10.1016/j.tcs.2015.10.011.
Cited on page 145.

[BRR17] Thomas Bläsius, Marcel Radermacher, and Ignaz Rutter. **How to Draw a Planarization**. In *Proceedings of the 43rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'17)*. Ed. by Bernhard Steffen, Christel Baier, Mark van den Brand, Johann Eder, Mike Hinchey, and Tiziana Margaria. Volume 10139 of Lecture Notes in Computer Science, pages 295–308. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-51963-0_23.
Cited on pages 17, 69.

[BRR19] Thomas Bläsius, Marcel Radermacher, and Ignaz Rutter. **How to Draw a Planarization**. *Journal of Graph Algorithms and Applications* 23:4, pages 653–682, 2019. DOI: 10.7155/jgaa.00506.
Cited on pages 17, 69.

[Buc+13] Christoph Buchheim, Markus Chimani, Carsten Gutwenger, Michael Jünger, and Petra Mutzel. **Crossings and Planarization**. In *Handbook of Graph Drawing and Visualization*. Ed. by Roberto Tamassia. Chapman and Hall/CRC, 2013. Chap. 2, pages 43–85.
Cited on pages 4, 5, 23, 24, 49, 70.

[CDR18] Markus Chimani, Hanna Döring, and Matthias Reitzner. **Crossing Numbers and Stress of Random Graphs**. In *Proceedings of the 26th International Symposium on Graph Drawing (GD'18)*. Ed. by Therese Biedl and Andreas Kerren. Volume 11282 of Lecture Notes in Computer Science, pages 255–268. Springer International Publishing, 2018. DOI: 10.1007/978-3-030-04414-5_18.
Cited on page 2.

[CH16] Markus Chimani and Petr Hlinený. **Inserting Multiple Edges into a Planar Graph**. In *Proceedings of the 32nd Annual Symposium on Computational Geometry (SoCG'16)*. Ed. by Sándor Fekete and Anna Lubiw. Volume 51 of Leibniz International Proceedings in Informatics (LIPIcs), pages 30:1–30:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. DOI: 10.4230/LIPIcs.SoCG.2016.30.
Cited on pages 23, 24, 49, 121.

[Cha+12] Erin W. Chambers, David Eppstein, Michael T. Goodrich, and Maarten Löffler. **Drawing Graphs in the Plane with a Prescribed Outer Face and Polynomial Area**. *Journal of Graph Algorithms and Applications* 16:2, pages 243–259, 2012. DOI: 10.7155/jgaa.00257.
Cited on page 69.

[Cha+15] Timothy M. Chan, Fabrizio Frati, Carsten Gutwenger, Anna Lubiw, Petra Mutzel, and Marcus Schaefer. **Drawing Partially Embedded and Simultaneously Planar Graphs**. *Journal of Graph Algorithms and Applications* 19:2, pages 681–706, 2015. DOI: 10.7155/jgaa.00375.
Cited on page 70.

[Cha+16] Steven Chaplick, Krzysztof Fleszar, Fabian Lipp, Alexander Ravsky, Oleg Verbitsky, and Alexander Wolff. **Drawing Graphs on Few Lines and Few Planes**. In *Proceedings of the 24th International Symposium on Graph Drawing (GD'16)*. Ed. by Yifan Hu and Martin Nöllenburg, pages 166–180. 2016. DOI: 10.1007/978-3-319-50106-2_14.
Cited on page 171.

[Cha+17] Steven Chaplick, Krzysztof Fleszar, Fabian Lipp, Alexander Ravsky, Oleg Verbitsky, and Alexander Wolff. **The Complexity of Drawing Graphs on Few Lines and Few Planes**. In *Proceedings of the 15th International Symposium on Algorithms and Data Structures (WADS'17)*. Ed. by Faith Ellen, Antonina Kolokolova, and Jörg-Rüdiger Sack, pages 265–276. 2017. DOI: 10.1007/978-3-319-62127-2_23.
Cited on page 171.

[Chi+09] Markus Chimani, Carsten Gutwenger, Petra Mutzel, and Christian Wolf. **Inserting a Vertex into a Planar Graph**. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'09)*, pages 375–383, 2009.
Cited on page 121.

[Chi+13] Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. **The Open Graph Drawing Framework (OGDF)**. In *Handbook of Graph Drawing and Visualization*. Ed. by Roberto Tamassia. Chapman and Hall/CRC, 2013. Chap. 17, pages 543–569.
Cited on pages 4, 23, 24, 55, 61, 81, 102, 103.

[CHW18] Markus Chimani, Ivo Hedtke, and Tilo Wiedera. **Exact Algorithms for the Maximum Planar Subgraph Problem: New Models and Experiments**. In *Proceedings of the 17th International Symposium on Experimental Algorithms (SEA'18)*. Ed. by Gianlorenzo D'Angelo. Volume 103 of Leibniz International Proceedings in Informatics (LIPIcs), pages 22:1–22:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. DOI: 10.4230/LIPIcs.SEA.2018.22.
Cited on page 26.

[CTU14] Javier Cano, Csaba D. Tóth, and Jorge Urrutia. **Upper Bound Constructions for Untangling Planar Geometric Graphs**. *SIAM Journal on Discrete Mathematics* 28:4, pages 1935–1943, 2014. DOI: 10.1137/130924172.
Cited on page 170.

[Cyg+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. **Parameterized Algorithms**. Springer International Publishing, 2015. ISBN: 978-3-319-21274-6. DOI: 10.1007/978-3-319-21275-3.
Cited on pages 140, 141.

[Da +18] Giordano Da Lozzo, Vida Dujmovic, Fabrizio Frati, Tamara Mchedlidze, and Vincenzo Roselli. **Drawing Planar Graphs with Many Collinear Vertices**. *Journal of Computational Geometry* 9:1, pages 94–130, 2018. DOI: 10.20382/jocg.v9i1a4.
Cited on pages ix, 3, 7, 69, 124, 169, 170, 171, 175.

[DEL11] Walter Didimo, Peter Eades, and Giuseppe Liotta. **Drawing Graphs with Right Angle Crossings**. *Theoretical Computer Science* 412:39, pages 5156–5166, 2011. DOI: 10.1016/j.tcs.2011.05.025.
Cited on page 97.

[Dem+18] Almut Demel, Dominik Dürrschnabel, Tamara Mchedlidze, Marcel Radermacher, and Lasse Wulf. **A Greedy Heuristic for Crossing-Angle Maximization**. In *Proceedings of the 26th International Symposium on Graph Drawing (GD'18)*. Ed. by Therese Biedl and Andreas Kerren. Volume 11282 of Lecture Notes in Computer Science, pages 286–299. Springer International Publishing, 2018. DOI: 10.1007/978-3-030-04414-5_20.
Cited on pages 17, 97.

[Dev+18] William Devanny, Phillip Kindermann, Maarten Löffler, and Ignaz Rutter. **Graph Drawing Contest Report**. In *Proceedings of the 25 International Symposium on Graph Drawing (GD'17)*. Ed. by Fabrizio Frati and Kwan-Liu Ma. Lecture Notes in Computer Science, pages 575–582. Springer International Publishing, 2018.
Cited on pages viii, 6, 97.

[DH11] Timothy A. Davis and Yifan Hu. **The University of Florida Sparse Matrix Collection** 38:1, pages 1:1–1:25, 2011. DOI: 10.1145/2049662.2049663.
Cited on page 61.

[DH96] Ron Davidson and David Harel. **Drawing Graphs Nicely Using Simulated Annealing**. *ACM Transactions on Graphics* 15:4, pages 301–331, 1996. DOI: 10.1145/234535.234538.
Cited on pages 24, 34.

[DL13a] Walter Didimo and Giuseppe Liotta. **The Crossing-Angle Resolution in Graph Drawing**. In *Thirty Essays on Geometric Graph Theory*. Ed. by Janos Pach. Springer New York, 2013, pages 167–184. DOI: 10.1007/978-1-4614-0110-0_10.
Cited on page 98.

[DL13b] Vida Dujmović and Stefan Langerman. **A Center Transversal Theorem for Hyperplanes and Applications to Graph Drawing**. *Discrete & Computational Geometry* 49:1, 2013. DOI: 10.1007/s00454-012-9464-y.
Cited on page 171.

[DLM19] Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. **A Survey on Graph Drawing Beyond Planarity** 52:1, pages 4:1–4:37, 2019. DOI: 10.1145/3301281.
Cited on page 70.

[DLR11] Walter Didimo, Giuseppe Liotta, and Salvatore A. Romeo. **Topology-Driven Force-Directed Algorithms**. In *Proceedings of the 18th International Symposium on Graph Drawing (GD'10)*. Ed. by Ulrik Brandes and Sabine Cornelsen. Volume 6502 of Lecture Notes in Computer Science, pages 165–176. Springer, 2011. DOI: 10.1007/978-3-642-18469-7_15.
Cited on page 70.

[DMW09] Tim Dwyer, Kim Marriott, and Michael Wybrow. **Topology Preserving Constrained Graph Layout**. In *Proceedings of the 16th International Symposium on Graph Drawing (GD'08)*. Ed. by Ioannis G. Tollis and Maurizio Patrignani. Volume 5417 of Lecture Notes in Computer Science, pages 230–241. Springer Berlin/Heidelberg, 2009. DOI: 10.1007/978-3-642-00219-9_22.
Cited on page 70.

[Duj+10] Vida Dujmović, Joachim Gudmundsson, Pat Morin, and Thomas Wolle. **Notes on Large Angle Crossing Graphs**. In *Proceedings of the 16th Symposium on Computing: The Australasian Theory (CATS'10)*. Ed. by Taso Viglas and Alex Potanin, pages 19–24. Australian Computer Society, 2010.
Cited on page 98.

[Duj+11] Vida Dujmović, William Evans, Stephen Kobourov, Giuseppe Liotta, Christophe Weibel, and Stephen Wismath. **On Graphs Supported by Line Sets**. In *Proceedings of the 18th International Symposium on Graph Drawing (GD'10)*. Ed. by Ulrik Brandes and Sabine Cornelsen, pages 177–182. Springer Berlin/Heidelberg, 2011. DOI: 10.1007/978-3-642-18469-7_16.
Cited on page 171.

[Duj+19] Vida Dujmović, Fabrizio Frati, Daniel Gonçalves, Pat Morin, and Günter Rote. **Every Collinear Set in a Planar Graph Is Free**. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*. Ed. by Timothy M. Chan, pages 1521–1538. Society for Industrial and Applied Mathematics, 2019. DOI: 10.1137/1.9781611975482.92.
Cited on page 3.

[Duj17] Vida Dujmović. **The Utility of Untangling**. *Journal of Graph Algorithms and Applications* 21:1, pages 121–134, 2017. DOI: 10.7155/jgaa.00407.
Cited on page 171.

[Ead+06] Peter Eades, Qingwen Feng, Xuemin Lin, and Hiroshi Nagamochi. **Straight-Line Drawing Algorithms for Hierarchical Graphs and Clustered Graphs**. *Algorithmica* 44:1, pages 1–32, 2006. DOI: 10.1007/s00453-004-1144-8.
Cited on page 145.

[Ead+15] Peter Eades, Seok-Hee Hong, Giuseppe Liotta, Naoki Katoh, and Sheung-Hung Poon. **Straight-Line Drawability of a Planar Graph Plus an Edge**. In *Proceedings of the 14th International Symposium on Algorithms and Data Structures (WADS'15)*. Ed. by Frank Dehne, Jörg-Rüdiger Sack, and Ulrike Stege. Volume 9214 of Lecture Notes in Computer Science, pages 301–313. Springer International Publishing, 2015. DOI: 0.1007/978-3-319-21840-3_25.
Cited on pages 6, 70, 121, 122, 123.

[Ead84] Peter Eades. **A Heuristic for Graph Drawing**. *Congressus Numerantium* 42, pages 149–160, 1984.
Cited on page 98.

[Eil98] Tali Eilam-Tzoreff. **The Disjoint Shortest Paths Problem**. *Discrete Applied Mathematics* 85:2, pages 113–138, 1998. DOI: 10.1016/S0166-218X(97)00121-2.
Cited on pages 133, 141, 206.

[EIS75] S. Even, A. Itai, and A. Shamir. **On the Complexity of Time Table and Multi-Commodity Flow Problems**. In *16th Annual Symposium on Foundations of Computer Science (SFCS 1975)*, pages 184–193, 1975. DOI: 10.1109/SFCS.1975.21.
Cited on page 133.

[Fár48] István Fáry. **On straight line representation of planar graphs**. *Acta Univ. Szeged. Sect. Sci. Math.* 11, pages 229–233, 1948.
Cited on pages viii, 2.

[FCE95] Qing-Wen Feng, Robert F. Cohen, and Peter Eades. **Planarity for Clustered Graphs**. In *Proceedings of the Third Annual European Symposium on Algorithms (ESA'95)*. Ed. by Paul Spirakis. Volume 979 of Lecture Notes in Computer Science, pages 213–226. Springer Berlin/Heidelberg, 1995. DOI: 10.1007/3-540-60313-1_145.
Cited on pages 144, 148.

[FHW80] Steven Fortune, John Hopcroft, and James Wyllie. **The Directed Subgraph Homeomorphism Problem**. *Theory of Computing Systems* 10:2, pages 111–121, 1980. DOI: 10.1016/0304-3975(80)90009-2.
Cited on page 133.

[FK97] Ulrich Fößmeier and Michael Kaufmann. **Nice Drawings for Planar Bipartite Graphs**. In *Algorithms and Complexity*. Ed. by Giancarlo Bongiovanni, Daniel Pierre Bovet, and Giuseppe Di Battista, pages 122–134. Springer Berlin/Heidelberg, 1997. DOI: 10.1007/3-540-62592-5_66.
Cited on page 175.

[FL14] Ruy Fabila-Monroy and Jorge López. **Computational Search of Small Point Sets with Small Rectilinear Crossing Number**. *Journal of Graph Algorithms and Applications* 18:3, pages 393–399, 2014. DOI: 10.7155/jgaa.00328.
Cited on pages 24, 50.

[FLM95] Arne Frick, Andreas Ludwig, and Heiko Mehldau. **A Fast Adaptive Layout Algorithm for Undirected Graphs (Extended Abstract and System Demonstration)**. In *Proceedings of the 2nd International Symposium on Graph Drawing (GD'94)*. Ed. by Roberto Tamassia and Ioannis G. Tollis, pages 388–403. Springer Berlin/Heidelberg, 1995. DOI: 10.1007/3-540-58950-3_393.
Cited on page 34.

[FPS16] Jacob Fox, János Pach, and Andrew Suk. **Approximating the Rectilinear Crossing Number**. In *Proceedings of the 24th International Symposium on Graph Drawing (GD'16)*. Ed. by Yifan Hu and Martin Nöllenburg. Lecture Notes in Computer Science, pages 413–426. Springer Berlin/Heidelberg, 2016. DOI: 10.1007/978-3-319-50106-2_32.
Cited on page 24.

[FR91] Thomas M.J. Fruchterman and Edward M. Reingold. **Graph Drawing by Force-Directed Placement**. *Software: Practice and Experience* 21:11, pages 1129–1164, 1991. DOI: 10.1002/spe.4380211102.
Cited on pages 34, 98, 103.

[Gia+12] Emilio Di Giacomo, Walter Didimo, Peter Eades, Seok-Hee Hong, and Giuseppe Liotta. **Bounds on the Crossing Resolution of Complete Geometric Graphs**. *Discrete Applied Mathematics* 160:1, pages 132–139, 2012. DOI: 10.1016/j.dam.2011.09.016.
Cited on page 97.

[Gia+18] Emilio Di Giacomo, Peter Eades, Giuseppe Liotta, Henk Meijer, and Fabrizio Montecchiani. **Polyline Drawings with Topological Constraints**. In *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC'18)*. Ed. by Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao. Volume 123 of Leibniz International Proceedings in Informatics (LIPIcs), pages 39:1–39:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. DOI: 10.4230/LIPIcs.ISAAC.2018.39.
Cited on page 70.

[GJ79] Michael R. Garey and David S. Johnson. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. W. H. Freeman, 1979. ISBN: 0-7167-1044-7.
Cited on pages vii, 26.

[GJ83] Michael R. Garey and David S. Johnson. **Crossing Number is NP-Complete**. *SIAM Journal on Algebraic and Discrete Methods* 4:3, pages 312–316, 1983.
Cited on pages 4, 24, 49, 121.

[GJT76] Michael R. Garey, David S. Johnson, and Robert Tarjan. **The Planar Hamiltonian Circuit Problem is NP-Complete**. *SIAM Journal on Computing* 5:4, pages 704–714, 1976. DOI: 10.1137/0205049.
Cited on page 176.

[GKM08] Carsten Gutwenger, Karsten Klein, and Petra Mutzel. **Planarity Testing and Optimal Edge Insertion with Embedding Constraints**. *Journal of Graph Algorithms and Applications* 12:1, pages 73–95, 2008. DOI: 10.7155/jgaa.00160.
Cited on page 121.

[GKN05] Emden R. Gansner, Yehuda Koren, and Stephen North. **Graph Drawing by Stress Majorization**. In *Proceedings of the 12th International Symposium on Graph Drawing (GD'04)*. Ed. by János Pach. Lecture Notes in Computer Science, pages 239–250. Springer Berlin/Heidelberg, 2005. DOI: 10.1007/978-3-540-31843-9_25.
Cited on pages 32, 34, 50, 55, 103.

[GMW05] Carsten Gutwenger, Petra Mutzel, and René Weiskircher. **Inserting an Edge Into a Planar Graph**. *Algorithmica* 41:4, pages 289–308, 2005. DOI: 10.1007/s00453-004-1128-8.
Cited on pages 26, 49, 85, 121.

[God95] Michael Godau. **On the Difficulty of Embedding Planar Graphs with Inaccuracies**. In *Proceedings of the Second International Symposium on Graph Drawing (GD'94)*. Ed. by Roberto Tamassia and Ioannis G. Tollis. Volume 894 of Lecture Notes in Computer Science, pages 254–261. Springer Berlin/Heidelberg, 1995.
Cited on page 144.

[Gri+14] Luca Grilli, Seok-Hee Hong, Jan Kratochvíl, and Ignaz Rutter. **Drawing Simultaneously Embedded Graphs with Few Bends**. In *Proceedings of the 22nd International Symposium on Graph Drawing (GD'14)*. Ed. by Christian Duncan and Antonios Symvonis. Volume 8871 of Lecture Notes in Computer Science, pages 40–51. Springer Berlin/Heidelberg, 2014. DOI: 10.1007/978-3-662-45803-7_4.
Cited on page 70.

[HH10] Weidong Huang and Maoling Huang. **Exploring the Relative Importance of Crossing Number and Crossing Angle**. In *Proceedings of the 3rd International Symposium on Visual Information Communication (VINCI '10)*. Ed. by Hongan Wang, Xiaoru Yuan, Linmi Tao, and Wei Chen, pages 10:1–10:8. ACM, 2010. DOI: 10.1145/1865841.1865854.
Cited on page 109.

[HH11] Stefan Huber and Martin Held. **Motorcycle graphs: Stochastic Properties Motivate an Efficient yet Simple Implementation**. *Journal of Experimental Algorithmics* 16, pages 1–3, 2011. DOI: 10.1145/1963190.2019578.
Cited on pages 72, 85.

[HH12] Stefan Huber and Martin Held. **A Fast Straight-Skeleton Algorithm Based on Generalized Motorcycle Graphs**. *International Journal of Computational Geometry & Applications* 22:05, pages 471–498, 2012. DOI: 10.1142/S0218195912500124.
Cited on page 85.

[HJ05] Stefan Hachul and Michael Jünger. **Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm**. In *Proceedings of the 12th International Symposium on Graph Drawing (GD'04)*. Ed. by János Pach, pages 285–295. Springer Berlin/Heidelberg, 2005. DOI: 10.1007/978-3-540-31843-9_29.
Cited on page 34.

[Hon+12] Seok-Hee Hong, Peter Eades, Giuseppe Liotta, and Sheung-Hung Poon. **Fáry's Theorem for 1-Planar Graphs**. In *Proceedings of the 18th Annual International Conference on Computing and Combinatorics (COCOON'12)*. Ed. by Joachim Gudmundsson, Julián Mestre, and Taso Viglas. Volume 7434 of Lecture Notes in Computer Science, pages 335–346. Springer Berlin/Heidelberg, 2012.
Cited on page 205.

[HS11] Sariel Har-Peled and Micha Sharir. **Relative (p, ϵ)-Approximations in Geometry**. *Discrete & Computational Geometry* 45:3, pages 462–496, 2011. DOI: 10.1007/s00454-010-9248-1.
Cited on page 58.

[Hua+10] Weidong Huang, Peter Eades, Seok-Hee Hong, and Chun- Cheng Lin. **Improving Force-Directed Graph Drawings by Making Compromises Between Aesthetics**. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'2010)*. Ed. by Christopher D. Hundhausen, Emmanuel Pietriga, Paloma Díaz, and Mary Beth Rosson, pages 176–183. IEEE Computer Society, 2010. DOI: 10.1109/VLHCC.2010.32.
Cited on pages 98, 99.

[JLM98] Michael Jünger, Sebastian Leipert, and Petra Mutzel. **A Note on Computing a Maximal Planar Subgraph using PQ-Trees**. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 17:7, pages 609–612, 1998. DOI: 10.1109/43.709399.
Cited on pages 26, 39.

[Jon17] Jonathan Klawitter and Tamara Mchedlidze and Martin Nöllenburg. **Experimental Evaluation of Book Drawing Algorithms**. In *Proceedings of the 25 International Symposium on Graph Drawing (GD'16)*. Ed. by Fabrizio Frati and Kwan-Liu Ma, pages 224–238, 2017. DOI: 10.1007/978-3-319-73915-1_19.
Cited on page 103.

[JS87] Barry Joe and Richard B. Simpson. **Corrections to Lee's Visibility Polygon Algorithm**. *BIT Numerical Mathematics* 27:4, pages 458–473, 1987.
Cited on page 75.

[Kan96] Goos Kant. **Drawing Planar Graphs Using the Canonical Ordering**. *Algorithmica* 16:1, pages 4–32, 1996. DOI: 10.1007/BF02086606.
Cited on pages 81, 85.

[KD79] M. S. Krishnamoorthy and Narsingh Deo. **Node-Deletion NP-Complete Problems**. *SIAM Journal on Computing* 8:4, pages 619–625, 1979. DOI: 10.1137/0208049.
Cited on page 26.

[KK89] Tomihisa Kamada and Satoru Kawai. **An Algorithm for Drawing General Undirected Graphs**. *Information Processing Letters* 31:1, pages 7–15, 1989. DOI: 10.1016/0020-0190(89)90102-6.
Cited on page 34.

[Kob13] Stephen G. Kobourov. **Force-Directed Drawing Algorithms**. In *Handbook of Graph Drawing and Visualization*. Ed. by Roberto Tamassia. Chapman and Hall/CRC, 2013. Chap. 12.
Cited on pages vii, 2, 4, 24, 50, 122.

[KS10] Yusuke Kobayashi and Christian Sommer. **On Shortest Disjoint Paths in Planar Graphs**. *Discrete Optimization* 7:4, pages 234–245, 2010. DOI: 10.1016/j.disopt.2010.05.002.
Cited on page 133.

[Lev26] Friedrich Levi. **Die Teilung der Projektiven Ebene durch Geraden oder Pseudogerade**. *Berichte Mathematische-Physikalischer Klassen der Sächsischen Akademie der Wissenschaften Leipzig* 78, pages 256–267, 1926.
Cited on page 171.

[LFF08] Andrea Lancichinetti, Santo Fortunato, and Filippo Filippo Radicchi. **Benchmark Graphs for Testing Community Detection Algorithms**. *Physical review E* 78:4, page 046110, 2008. DOI: 10.1103/PhysRevE.78.046110.
Cited on pages 32, 102.

[LLS01] Yi Li, Philip M. Long, and Aravind Srinivasan. **Improved Bounds on the Sample Complexity of Learning**. *Journal of Computer and System Sciences* 62:3, pages 516–527, 2001. DOI: 10.1006/jcss.2000.1741.
Cited on page 58.

[LY80] John M. Lewis and Mihalis Yannakakis. **The Node-Deletion Problem for Hereditary Properties is NP-Complete**. *Journal of Computer and System Sciences* 20:2, pages 219–230, 1980. DOI: http://dx.doi.org/10.1016/0022-0000(80)90060-4.
Cited on page 26.

[Mat02] Jiří Matoušek. **Lectures on Discrete Geometry**. 1st ed. Volume 212 of Graduate Texts in Mathematics. Springer New York, 2002. DOI: 10.1007/978-1-4613-0039-7.
Cited on page 56.

[Mch+19a] Tamara Mchedlidze, Marcel Radermacher, Ignaz Rutter, and Peter Stumpf. **Stretching Two Pseudolines in Planar Straight-Line Drawings**. In *Proceedings of the 27th International Symposium on Graph Drawing (GD'19)*. Ed. by Daniel Archambault and Csaba D. Tóth. Lecture Notes in Computer Science. Springer International Publishing, 2019.
Cited on page 169.

[Mch+19b] Tamara Mchedlidze, Marcel Radermacher, Ignaz Rutter, and Nina Zimbel. **Drawing Clustered Graphs on Disk Arrangements**. *Journal of Graph Algorithms and Applications* 24:2, pages 105–131, 2019. DOI: 10.7155/jgaa.00521.
Cited on page 143.

[Mch+19c] Tamara Mchedlidze, Marcel Radermacher, Ignaz Rutter, and Nina Zimbel. **Drawing Clustered Graphs on Disk Arrangements**. In *Proceedings of the 13th International Conference and Workshops on Algorithms and Computation (WALCOM '19)*. Volume 11355 of Lecture Notes in Computer Science. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-10564-8_13.
Cited on page 143.

[Mnë88] Nikolai E. Mnëv. **The Universality Theorems on the Classification Problem of Configuration Varieties and Convex Polytopes Varieties**. In *Topology and Geometry — Rohlin Seminar*. Ed. by Oleg Yanovich Viro and Anatoly Moiseevich Vershik. Volume 1346. Lecture Notes in Mathematics. Springer Berlin/Heidelberg, 1988, pages 527–543. DOI: 10.1007/BFb0082792.
Cited on pages 2, 70, 171.

[MNR16] Tamara Mchedlidze, Martin Nöllenburg, and Ignaz Rutter. **Extending Convex Partial Drawings of Graphs**. *Algorithmica* 76:1, pages 47–67, 2016. DOI: 10.1007/s00453-015-0018-6.
Cited on pages viii, 3, 70.

[MNS18] Henning Meyerhenke, Martin Nöllenburg, and Christian Schulz. **Drawing Large Graphs by Multilevel Maxent-Stress Optimization** 24:5, pages 1814–1827, 2018. DOI: 10.1109/TVCG.2017.2689016.
Cited on pages 50, 61.

[Moo91] Thomas L. Moore. **Using Euler's Formula to Solve Plane Separation Problems**. *The College Mathematics Journal* 22:2, pages 125–130, 1991.
Cited on page 59.

[MRR18a] Tamara Mchedlidze, Marcel Radermacher, and Ignaz Rutter. **Aligned Drawings of Planar Graphs**. *Journal of Graph Algorithms and Applications* 22:3, pages 401–429, 2018. DOI: 10.7155/jgaa.00475.
Cited on page 169.

[MRR18b] Tamara Mchedlidze, Marcel Radermacher, and Ignaz Rutter. **Aligned Drawings of Planar Graphs**. In *Proceedings of the 25th International Symposium on Graph Drawing (GD'17)*. Ed. by Fabrizio Frati and Kwan-Liu Ma. Volume 10692 of Lecture Notes in Computer Science, pages 3–16. Springer International Publishing, 2018. DOI: 10.1007/978-3-319-73915-1_1.
Cited on page 169.

[NP82] Jürg Nievergelt and Franco P. Preparata. **Plane-Sweep Algorithms for Intersecting Geometric Figures**. *Communications of the ACM* 25:10, pages 739–747, 1982. DOI: 10.1145/358656.358681.
Cited on pages 75, 79.

[P80] Jacob E. Goodman . and Richard Pollack. **Proof of Grünbaum's Conjecture on the Stretchability of Certain Arrangements of Pseudolines**. *Journal of Combinatorial Theory, Series A* 29:3, pages 385–390, 1980. DOI: 10.1016/0097-3165(80)90038-2.
Cited on page 171.

[PCJ96] Helen C. Purchase, Robert F. Cohen, and Murray James. **Validating Graph Drawing Aesthetics**. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*. Ed. by Franz J. Brandenburg, pages 435–446. Springer Berlin/Heidelberg, 1996. DOI: 10.1007/BFb0021827.
Cited on pages 2, 23.

[Pur97] Helen Purchase. **Which Aesthetic has the Greatest Effect on Human Understanding?** In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Volume 1353 of Lecture Notes in Computer Science, pages 248–261. Springer Berlin/Heidelberg, 1997. DOI: 10.1007/3-540-63938-1_67.
Cited on page 2.

[Rad+18] Marcel Radermacher, Klara Reichard, Ignaz Rutter, and Dorothea Wagner. **A Geometric Heuristic for Rectilinear Crossing Minimization**. In *Proceedings of the 20th Workshop on Algorithm Engineering and Experiments (ALENEX'18)*, pages 129–138, 2018. DOI: 10.1137/1.9781611975055.12.
Cited on pages 17, 23, 40.

[Rad+19] Marcel Radermacher, Klara Reichard, Ignaz Rutter, and Dorothea Wagner. **Geometric Heuristics for Rectilinear Crossing Minimization**. *Journal of Experimental Algorithmics* 24:1, pages 1.12:1–1.12:21, 2019. DOI: 10.1145/3325861.
Cited on pages 17, 23.

[Rad15] Marcel Radermacher. **How to Draw a Planarization**. Master Thesis. Karlsruhe Institute of Technology, Institute of Theoretical Informatics, 2015. URL: https://i11www.iti.kit.edu/_media/teaching/theses/ma-radermacher-15.pdf.
Cited on pages 17, 69.

[Rei16] Klara Reichard. **Rectilinear Crossing Minimization**. Master Thesis. Karlsruhe Institute of Technology, Institute of Theoretical Informatics, 2016. URL: https://i11www.iti.kit.edu/_media/teaching/theses/ma-reichard-16.pdf.
Cited on page 23.

[Rib06] Ares Ribó Mor. **Realization and Counting Problems for Planar Structures**. PhD thesis. FU Berlin, 2006. URL: https://refubium.fu-berlin.de/handle/fub188/10243.
Cited on page 145.

[RR18] Marcel Radermacher and Ignaz Rutter. **Inserting an Edge into a Geometric Embedding**. In *Proceedings of the 26th International Symposium on Graph Drawing (GD'18)*. Ed. by Therese C. Biedl and Andreas Kerren. Volume 11282 of Lecture Notes in Computer Science, pages 402–415. Springer International Publishing, 2018. DOI: 10.1007/978-3-030-04414-5_29.
Cited on page 121.

[RR19] Marcel Radermacher and Ignaz Rutter. **Geometric Crossing Minimization - A Scalable Randomized Approach**. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA'19)*. Ed. by Michael A. Bender, Ola Svensson, and Grzegorz Herman. Leibniz International Proceedings in Informatics (LIPIcs), pages 76:1–76:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. DOI: 10.4230/LIPIcs.ESA.2019.76.
Cited on page 49.

[RV11] Alexander Ravsky and Oleg Verbitsky. **On Collinear Sets in Straight-Line Drawings**. In *Proceedings of the 37th Workshop on Graph-Theoretic Concepts in Computer Science (WG'11)*. Ed. by Petr Kolman and Jan Kratochvíl, pages 295–306. 2011.
Cited on page 171.

[Sch10] Marcus Schaefer. **Complexity of Some Geometric and Topological Problems**. In *Proceedings of the 17th International Symposium on Graph Drawing (GD'09)*. Ed. by David Eppstein and Emden R. Gansner. Volume 5849 of Lecture Notes in Computer Science, pages 334–344. Springer Berlin/Heidelberg, 2010. DOI: 10.1007/978-3-642-11805-0_32.
Cited on pages 9, 24, 69.

[She03] David J Sheskin. **Handbook of Parametric and Nonparametric Statistical Procedures**. Chapman and Hall/CRC, 2003.
Cited on pages 19, 20, 65.

[Sho91] Peter W. Shor. **Stretchability of Pseudolines is NP-Hard**. In *Applied Geometry and Discrete Mathematics–The Victor Klee Festschrift*. Ed. by Bernd Sturmfels and Peter Gritzmann. Volume 4. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. AMS, DIMACS and ACM, 1991, pages 531–554.
Cited on pages 2, 24, 27, 171.

[Sim+11] Paolo Simonetto, Daniel Archambault, David Auber, and Romain Bourqui. **ImPrEd: An Improved Force-Directed Algorithm that Prevents Nodes from Crossing Edges**. *Computer Graphics Forum* 30:3, pages 1071–1080, 2011. DOI: 10.1111/j.1467-8659.2011.01956.x.
Cited on pages 69, 70, 72, 73.

[SSM16] Christian L. Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. **NetworKit: A Tool Suite for Large-Scale Complex Network Analysis**. *Network Science* 4:4, pages 508–530, 2016. DOI: 10.1017/nws.2016.20.
Cited on pages 32, 102.

[Ste51] Sherman K. Stein. **Convex maps**. *Proceedings of the American Mathematical Society* 2, pages 464–466, 1951. DOI: 10.2307/2031777.
Cited on page 2.

[SW99] Angelika Steger and Nicholas C. Wormald. **Generating Random Regular Graphs Quickly**. *Combinatorics, Probability and Computing* 8:4, pages 377–396, 1999. DOI: /10.1017/S0963548399003867.
Cited on page 61.

[Tam87] Roberto Tamassia. **On Embedding a Graph in the Grid with the Minimum Number of Bends**. *SIAM Journal on Computing* 16:3, pages 421–444, 1987. DOI: 10.1137/0216030.
Cited on page 70.

[The17] The CGAL Project. **CGAL User and Reference Manual (http://doc.cgal.org/4.10/Manual/packages.html)**. 4.10. CGAL Editorial Board, 2017.
Cited on pages 32, 52.

[Tho88] Carsten Thomassen. **Rectilinear Drawings of Graphs**. *Journal of Graph Theory* 12:3, pages 335–341, 1988. DOI: 10.1002/jgt.3190120306.
Cited on page 70.

[Tut63] William T. Tutte. **How to Draw a Graph**. *Proceedings of the London Mathematical Society* 3:1, pages 743–767, 1963. DOI: 10.1112/plms/s3-13.1.743.
Cited on pages viii, 3, 69, 174.

[VC71] Vladimir N. Vapnik and Alexey Y. Chervonenkis. **On the Uniform Convergence of Relative Frequencies of Event to their Probabilities**. *Theory of Probability & Its Application* 16:2, pages 264–280, 1971. DOI: 10.1137/1116025.
Cited on page 58.

[Vrt14] Imrich Vrt'o. **Bibliography on Crossing Numbers of Graphs. (ftp://ftp.ifi.savba.sk/pub/imrich/crobib.pdf)**. 2014.
Cited on pages vii, 2, 23, 49, 205.

[Wag36] Klaus Wagner. **Bemerkungen zum Vierfarbenproblem.** 46, pages 26–32, 1936.
Cited on page 2.

[Wah13] Magnus Wahlström. **Abusing the Tutte Matrix: An Algebraic Instance Compression for the K-set-cycle Problem**. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS '13)*. Ed. by Natacha Portier and Thomas Wilke, pages 341–352. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013. DOI: 10.4230/LIPIcs.STACS.2013.341.
Cited on page 180.

[Zim17] Nina Zimbel. **Unpacking Planar Clustered Graphs:To Bend or not to Bend?** Bachelor Thesis. Karlsruhe Institute of Technology, Institute of Theoretical Informatics, 2017. URL: https://i11www.iti.kit.edu/_media/teaching/theses/ba-zimbel-17.pdf.
Cited on page 143.